

HTML5与CSS3 实战指南

Alexis Goldstein
[美] Louis Lazaris 著
Estelle Weyl
宋松 译

人民邮电出版社
北 京

Java学习群：72030155

好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间：可以联系群主获取更多大型企业内部技术教程。

图书在版编目 (C I P) 数据

HTML5与CSS3实战指南 / (美) 古德斯特恩 (Goldstein, A.), (美) 拉扎里斯 (Lazaris, L.), (美) 威尔 (Weyl, E.) 著; 宋松译. — 北京: 人民邮电出版社, 2011.12
ISBN 978-7-115-26587-6

I. ①H… II. ①古… ②拉… ③威… ④宋… III. ①超文本标记语言, HTML 5—程序设计—指南②网页制作工具, CSS 3—指南 IV. ①TP312-62②TP393.092-62

中国版本图书馆CIP数据核字(2011)第205314号

版权声明

© PT Press 2011

Authorized translation of the English edition of HTML5 & CSS3 For The Real World, First Edition © 2011 SitePoint Pty. Ltd. This translation is published and sold by permission of O'Reilly Media, Inc., the owner of all rights to publish and sell the same.

All rights reserved.

本书中文简体字版由 O'Reilly Media, Inc. 授权人民邮电出版社出版。未经出版者书面许可, 对本书任何部分不得以任何方式复制或抄袭。

版权所有, 侵权必究。

HTML 5 与 CSS3 实战指南

- ◆ 著 [美] Alexis Goldstein Louis Lazaris Estelle Weyl
- 译 宋 松
- 责任编辑 傅道坤
- ◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子邮件 315@ptpress.com.cn
网址 <http://www.ptpress.com.cn>
北京鑫正大印刷有限公司印刷
- ◆ 开本: 800×1000 1/16
印张: 19.25
字数: 353 千字 2011 年 12 月第 1 版
印数: 1-4 000 册 2011 年 12 月北京第 1 次印刷

著作权合同登记号 图字: 01-2011-5252 号

ISBN 978-7-115-26587-6

定价: 45.00 元

读者服务热线: (010)67132705 印装质量热线: (010)67129223

反盗版热线: (010)67171154

内 容 提 要

本书主要介绍 HTML5 和 CSS3 的新功能，内容系统全面，易于理解。

本书由 11 章和 3 个附录组成，内容包括 HTML5 和 CSS3 简介、HTML5 标记、HTML5 语义、HTML5 表单、HTML5 音频和视频、CSS3 渐变和多背景、CSS 转换和过渡、嵌入字体和多列布局、地理定位、离线 Web 应用和 Web 存储、画布、SVG 和拖放、Modernizr、WAI-ARIA 以及微数据。每章都包含代码示例，供读者在学习过程中进行参考。

本书适合想要了解最新一代的基于浏览器技术的 Web 设计人员和前端开发人员阅读，是您了解最新一代的 Web 开发技术的理想书籍。

前 言

欢迎读者购买并阅读本书。我们很高兴您能加入我们的旅程，来探索最新的前端网站构建技术。

如果您选择了这本书，则表明您很可能对 HTML 和 CSS 有了一定的了解。甚至可能您已经是一个在某些领域（比如标记、样式或脚本）具有丰富经验的专业人员，而现在想要通过学习与 HTML5 和 CSS3 相关的新功能和技术，来进一步提高技能。

学习一项新任务可能很困难。您可能只有有限的时间来研究这些基于 Web 语言的官方文档和规范。而那些可以作为参考使用的大部头技术图书，则因为提供的实用案例较少，从而招致您的反感。

基于这个原因，本书的目标是帮助您学习实用的说明，从而帮助您解决目前在构建网站时遇到的实际问题。当然，本书主要介绍 HTML5 和 CSS3。

但是本书并不仅仅是一个分步教程。在本书中，我们提供了大量的理论和技术信息，帮助您弥补部分理解空白（新技术出现的原因及其工作原理），同时又尽力确保您不会被大量新知识而击垮。现在开始吧！

本书的读者对象

本书针对的是想要了解最新一代的基于浏览器技术的 Web 设计人员和前端开发人员。您应该至少已经具有 HTML 和 CSS 的一般知识，因为我们不会介绍关于标记和样式的基础知识，我们将着重介绍 HTML5 和 CSS3 所具有的新功能。

本书的最后两章介绍与 HTML5 相关的一些新 JavaScript API。当然，理解这两章需要您大致了解 JavaScript，但是这并不是了解本书其他内容所必备的。如果您不熟悉 JavaScript，现在跳过这两章，待以后熟悉了相关内容再返回来阅读它们也是可以的。

本书的内容

本书由 11 章和 3 个附录组成。大部分章节内容都是连续的，所以建议您按顺序阅读它们，以便获得最大受益，但是如果您只是想阅读特定主题的最新内容，当然也可以跳过一些内容。

第 1 章 HTML5 和 CSS3 简介

在我们介绍实际内容之前，先简单介绍一些历史，以及现在开始使用 HTML5 和 CSS3 的一些具有说服力的理由。我们将介绍目前的浏览器支持情况，同时我们认为：只要利用方式合理，大部分新技术目前都可以使用。

第 2 章 HTML5 样式的标记

本章介绍了 HTML5 中新增的一些新结构和语义元素。我们将介绍 The HTML5 Herald，它是本书使用的一个演示网站。您认为 divs 很枯燥？我们也是这么认为的。有关 HTML5 的一个好消息是，现在它提供了各种各样的选项：article、section、nav、footer、aside 和 header！

第 3 章 关于 HTML5 语义的更多内容

继续上一章的内容，我们将介绍 HTML5 构建文档大纲的新方式。然后介绍了许多其他的语义元素，以便您更熟悉标记。

第 4 章 HTML5 表单

目前 HTML5 中一些最有用和实用的功能是与表单相关的。目前许多浏览器都支持对电子邮件类型（比如电子邮件和 URL）进行本地验证，而一些浏览器甚至支持本地日期选择器、滑块和微调框。这几乎足以使您喜欢代码表单！本章涵盖了加快编写 HTML5 表单速度所需的一切内容，并针对旧版浏览器提供了脚本回调。

第 5 章 HTML5 音频和视频

HTML5 经常被吹捧为在线多媒体内容王冠的争夺者，而之前这一荣誉由 Flash 长期占有。新的 audio 和 video 元素是成为争夺者的原因，它们为媒体提供可编写脚本的本地容器，而不像 Flash 那样需要依赖第三方插件。在本章中，您将了解让这些元素工作的详细信息。

第 6 章 CSS3 简介

我们已经介绍了有关 HTML5 的所有内容，现在是时候介绍 CSS3 了。对于 CSS3，我们首先介绍一些新选择器，它们能够让您以前所未有的灵活性来定位页面上的元素。接着介绍在 CSS3 中指定颜色（包括透明）的一些新方式。本章最后介绍了几个快速获胜技巧，可以最少的工作将杰出的 CSS3 功能添加到您的网站：文本阴影、投影和圆角。

第 7 章 CSS3 渐变和多背景

您上次在没有渐变或背景图像的网站工作是什么时候？CSS3 为在 Photoshop 中花费大量时间来进行图片处理的开发人员提供了迟到的支持，尝试在不破坏带宽存储单元的情况下创建完美的背景渐变和图像。现在您可以在 CSS3 中指定线性或径向渐变，无需图像，并且您可以为一个元素指定任意数量的背景图像。现在是时候抛弃这些用起来很费劲的旧 div 了。

第 8 章 CSS 转换和过渡

动画早已被视为 JavaScript 的一个主要功能，但是 CSS3 可让您将一些困难的工作转嫁到浏览器上。转换可让您旋转、翻转、扭曲图像，或者是到处移动图像。过渡可为我们在网站上看到的不和谐的所有打开或所有关闭状态更改添加一些细微的差别。我们以对未来的简单介绍结束本章内容；尽管 CSS 关键帧动画仍然缺少广泛的支持，但我们还是觉得您会看好它们。

第 9 章 嵌入字体和多列布局

您喜欢 Arial 或 Verdana 字体，还是 Georgia 或 Times？如果这些字体都没有呢？在本章中，我们将介绍如何移动过去的“网络安全”字体，并将任意字体嵌入到我们的页面中，以供访问者在下载样式表和图像时同时下载字体。我们还将介绍一项大有可为的新 CSS 功能，它支持我们跨越多个列布局内容，而不需要使用额外标记或令人畏惧的 float。

第 10 章 地理定位、离线 Web 应用和 Web 存储

最新一代的浏览器都配备了大量新标准的 JavaScript API 选项。其中有许多是针对移动浏览器的，但仍可以供桌面用户使用。在本章中，我们将介绍最令人兴奋的 3 个功能：地理定位、离线 Web 应用和 Web 存储。我们也简单介绍了一些 API，在本书中不会详细介绍它们，因为它们的支持不足，或者是仅具备有限的用例。此外，我们还介绍了一些您想要继续研究的指针。

第 11 章 画布、SVG 和拖放

本书的最后一章首先介绍了两种可绘制和显示图形的具有竞争力的技术。`canvas`（下文中称之为画布）是 HTML5 的新增功能，并提供一个像素表面和一个 JavaScript API 来在画布上绘制形状。另一方面，SVG 已出现了多年，但是现在已获得了浏览器很好的支持，因此它正成为一种越来越可行的选择。最后，我们介绍了另一个新 JavaScript API，即拖放，它提供了本机拖放界面处理。

附录 A Modernizr

作为 HTML5 超级工具中的一个主要工具，Modernizr 是一个极好的小 JavaScript 库，可以检测到几乎所有的 HTML5 和 CSS3 功能支持，支持您选择性地设置网站样式，或者应用回调策略。在此附录中我们包含了一个如何使用 Modernizr 的快速入门书，Modernizr 的使用贯穿全书。这样，您可以将本附录作为一个现成的参考，其他章则主要介绍 HTML5 和 CSS3。

附录 B WAI-ARIA

WAI-ARIA 是一个独立的规范，通常与 HTML5 相提并论，它是最新的工具集，用来帮助使用辅助技术的用户访问成熟的网络应用程序。虽然可以整本书都用来介绍 WAI-ARIA，但是我们认为包含它的快速摘要和一些可了解更多内容的链接是有益的。

附录 C 微数据

微数据是 HTML5 规范的一部分，使用机器可读的标签来标注标记。它仍处于不断变化之中，但是我们认为它值得通过几个示例来帮助您了解它。

在哪里可以找到帮助

SitePoint 有一个活跃的网络设计人员和开发人员社区，如果您遇到问题，可以向他们寻求帮助。我们还有一个关于本书的已知勘误表列表，可以在那里查找最新的更新。

SitePoint 论坛

SitePoint 论坛¹是询问与任何网络开发相关的问题讨论论坛。当然，您也可以回

¹ <http://www.sitepoint.com/forums/>

答问题。这是论坛网站的工作方式：一些人询问，一些人回答，大多数人则两种事情都做。分享您的知识可以使其他人受益，并增强社区的实力。社区里有很多有趣和经验丰富的网络开发设计人员和开发人员。这是一种学习新知识并迅速解决问题的好方式。

本书的网站

本书的网址为 <http://sitepoint.com/books/rw1/>，可以在本网站访问下列内容：

代码文档

阅读本书时，您会注意到许多指向代码文档的引用出处。这是一个可下载的 ZIP 文档，包含本书中的所有示例源代码。如果您不想手动输入这些源代码（或者避免自己发生腕管综合症），可下载这些代码文档¹。

更新和勘误表

没有任何一本书是完美的，我们希望认真的读者能够至少发现一个或两个错误。本书网站上的勘误表页面²将包含有关已知印刷和代码错误的任何最新信息。

SitePoint 邮件列表

除了类似本书这样的书籍外，SitePoint 还发布了很多免费的电子邮件列表，比如 SitePoint Tech Times、SitePoint Tribune 和 SitePoint Design View 等。在这些邮件列表中，您将了解到网络开发的最新新闻、产品版本、趋势、提示和技术等内容。读者可注册一个或多个 SitePoint 邮件列表，网址为 <http://www.sitepoint.com/newsletter/>。

SitePoint 播客

加入 SitePoint 播客团队，了解关于网络开发人员和设计人员的新闻、采访、想法和最新思想。我们讨论了最新的网络行业主题，展示了嘉宾演讲者并采访了一些

¹ <http://www.sitepoint.com/books/rw1/code.php>

² <http://www.sitepoint.com/books/rw1/errata.php>

行业内的顶尖人物。您可以在 www.sitepoint.com/podcast/ 或者通过 iTunes 订阅来获得最新和之前的播客内容。

您的反馈

如果无法通过论坛找到答案，或者您出于其他原因想要联系我们，可在 books@sitepoint.com 撰写问题。我们有一个很好的电子邮件支持系统来跟踪问题，并且如果我们的支持团队成员无法回答您的问题，他们就会将问题直接发给我们。特别欢迎您向我们发送改进建议和发现的任何错误通知。

本书中使用的约定

您将注意到我们在整本书中使用了特定印刷和布局样式，以表示不同类型的信息。注意下列条目。

代码示例

本书中的代码使用固定宽度的字体显示：

```
<h1>A Perfect Summer's Day</h1>
<p>It was a lovely day for a walk in the park. The birds
were singing and the kids were all back at school.</p>
```

如果在本书的代码存档中找到该代码，则文件的名称将出现在程序列表的顶部：

```
example.css

.footer {
  background-color: #CCC;
  border-top: 1px solid #333;
}
```

如果只显示了文件的一部分，那么就用“excerpt”一词表示：

```
example.css (excerpt)

border-top: 1px solid #333;
```

若将其他代码插入到一个现有的示例中，则新代码将以粗体形式显示：

```
function animate() {
  new_variable = "Hello";
}
```

当上下文需要现有代码而不是重复所有代码时，将显示垂直省略号：

```
function animate() {
  :
  return new_variable;
}
```

一些代码行应输入到一行中，但由于页面限制我们只好换行。↵表示换行符，仅用于格式化目的，可忽略它：

```
URL.open("http://www.sitepoint.com/blogs/2007/05/28/user-style-sheets-come-of-age/");
```

提示、备注和警告



喂！

提示将为您提供一些有用的小线索。



嗨，抱歉……

备注是关于目前主题的一些有用的信息，但不一定是关键信息。可将这些信息看做额外的花絮信息。



确保您总是……

……注意这些要点。



小心

警告将突出您在阅读本书过程中可能会遇到的陷阱。

目 录

| | |
|-----------------------------------|----|
| 第 1 章 HTML5 和 CSS3 简介 | 1 |
| 1.1 什么是 HTML5 | 1 |
| 1.2 如何发展至今 | 2 |
| 1.3 我们为什么应关注 HTML5 | 4 |
| 1.4 什么是 CSS3 | 4 |
| 1.5 我们为什么应关注 CSS3 | 5 |
| 1.6 在现实中，我们想做的是 | 6 |
| 1.6.1 多种浏览器的市场 | 6 |
| 1.6.2 不断发展的移动市场 | 7 |
| 1.7 实际应用 | 8 |
| 第 2 章 HTML5 样式的标记 | 11 |
| 2.1 The HTML5 Herald 简介 | 11 |
| 2.2 基本的 HTML5 模板 | 12 |
| 2.2.1 Doctype | 13 |
| 2.2.2 html 元素 | 14 |
| 2.2.3 head 元素 | 14 |
| 2.2.4 公平竞争 | 16 |
| 2.2.5 剩余部分是历史简介 | 17 |
| 2.3 HTML5 常见问题 | 18 |
| 2.3.1 为什么这些变更仍能够在旧版浏览器中工作 | 18 |
| 2.3.2 是否需要关闭所有标签 | 20 |
| 2.3.3 关于其他的基于 XHTML 的语法使用习惯 | 20 |
| 2.4 定义页面结构 | 22 |
| 2.4.1 header 元素 | 22 |
| 2.4.2 section 元素 | 23 |
| 2.4.3 article 元素 | 24 |
| 2.4.4 nav 元素 | 25 |
| 2.4.5 aside 元素 | 26 |

| | | |
|-----------------------------------|---------------------------|-----------|
| 2.4.6 | footer元素 | 27 |
| 2.5 | 构建 The HTML5 Herald | 28 |
| 2.6 | 小结 | 30 |
| 第3章 关于 HTML5 语义的更多内容 | | 31 |
| 3.1 | 关于内容类型的新视角 | 31 |
| 3.2 | 文档大纲 | 33 |
| 3.3 | 最新消息 | 34 |
| 3.4 | 更多新元素 | 37 |
| 3.4.1 | figure和figcaption元素 | 37 |
| 3.4.2 | mark元素 | 38 |
| 3.4.3 | progress和meter元素 | 38 |
| 3.4.4 | time元素 | 39 |
| 3.5 | 对现有功能的更改 | 41 |
| 3.5.1 | 单词“Deprecated”是被弃用的 | 41 |
| 3.5.2 | 链接中的block元素 | 41 |
| 3.5.3 | 黑体文本 | 41 |
| 3.5.4 | 斜体文本 | 42 |
| 3.5.5 | 大号和小号文体 | 43 |
| 3.5.6 | 引起争议的cite元素 | 43 |
| 3.5.7 | 描述（不是定义）列表 | 43 |
| 3.6 | 其他新元素及功能 | 44 |
| 3.6.1 | details元素 | 44 |
| 3.6.2 | 自定义的有序列表 | 45 |
| 3.6.3 | 作用域样式 | 45 |
| 3.6.4 | script元素的async属性 | 45 |
| 3.7 | 验证 HTML5 文档 | 46 |
| 3.8 | 小结 | 48 |
| 第4章 HTML5 表单 | | 49 |
| 4.1 | 工具箱中的相关工具 | 49 |
| 4.2 | HTML5 表单属性 | 51 |
| 4.2.1 | required属性 | 51 |
| 4.2.2 | placeholder属性 | 55 |
| 4.2.3 | pattern属性 | 58 |
| 4.2.4 | disabled属性 | 59 |

| | | |
|--------|-------------------|----|
| 4.2.5 | readonly属性 | 59 |
| 4.2.6 | multiple属性 | 60 |
| 4.2.7 | form属性 | 60 |
| 4.2.8 | autocomplete属性 | 61 |
| 4.2.9 | datalist元素和list属性 | 61 |
| 4.2.10 | autofocus属性 | 62 |
| 4.3 | HTML5 新表单输入类型 | 62 |
| 4.3.1 | search | 63 |
| 4.3.2 | Email Addresses | 64 |
| 4.3.3 | URL | 65 |
| 4.3.4 | Telephone Numbers | 66 |
| 4.3.5 | Numbers | 66 |
| 4.3.6 | Ranges | 67 |
| 4.3.7 | Colors | 68 |
| 4.3.8 | Dates和Times | 69 |
| 4.4 | HTML5 中的其他新表单控件 | 71 |
| 4.4.1 | output元素 | 72 |
| 4.4.2 | keygen元素 | 72 |
| 4.5 | 对现有表单控件及属性的更改 | 72 |
| 4.5.1 | form元素 | 72 |
| 4.5.2 | optgroup元素 | 73 |
| 4.5.3 | textarea元素 | 73 |
| 4.6 | 小结 | 73 |
| 第 5 章 | HTML5 音频和视频 | 75 |
| 5.1 | 历史简介 | 75 |
| 5.2 | 目前状况 | 76 |
| 5.2.1 | 视频容器格式 | 76 |
| 5.2.2 | 视频编解码器 | 77 |
| 5.2.3 | 音频编解码器 | 77 |
| 5.2.4 | 当前浏览器使用哪种组合 | 77 |
| 5.3 | 标记 | 78 |
| 5.3.1 | 启用本机控件 | 78 |
| 5.3.2 | autoplay属性 | 79 |
| 5.3.3 | loop属性 | 80 |
| 5.3.4 | preload属性 | 80 |

| | | |
|--------|---------------------------|-----|
| 5.3.5 | poster属性 | 81 |
| 5.3.6 | audio属性 | 81 |
| 5.3.7 | 添加对多种视频格式的支持 | 81 |
| 5.3.8 | 资源顺序 | 82 |
| 5.3.9 | 关于Internet Explorer 6 ~ 8 | 83 |
| 5.3.10 | MIME类型 | 85 |
| 5.4 | 用于网络的视频文件解码 | 86 |
| 5.5 | 创建自定义控件 | 86 |
| 5.5.1 | 让我们从一些标记和设计开始 | 87 |
| 5.5.2 | 介绍媒体元素API | 88 |
| 5.5.3 | 播放和暂停视频 | 90 |
| 5.5.4 | 视频音轨的静音与取消静音 | 93 |
| 5.5.5 | 视频结束播放的响应 | 94 |
| 5.5.6 | 更新视频播放的时间 | 94 |
| 5.5.7 | 媒体元素API的其他一些功能 | 97 |
| 5.6 | 关于音频 | 99 |
| 5.7 | 可访问的媒体 | 99 |
| 5.8 | 小结 | 100 |
| 第6章 | CSS3 简介 | 101 |
| 6.1 | 改进旧版浏览器 | 101 |
| 6.2 | CSS3 选择器 | 102 |
| 6.2.1 | 关系选择器 | 102 |
| 6.2.2 | 属性选择器 | 104 |
| 6.2.3 | 伪类 | 105 |
| 6.2.4 | 结构化伪类 | 107 |
| 6.2.5 | 伪元素和生成的内容 | 110 |
| 6.3 | CSS3 颜色 | 111 |
| 6.3.1 | RGBA | 112 |
| 6.3.2 | HSL和HSLA | 113 |
| 6.3.3 | 不透明度 | 114 |
| 6.4 | 实际应用 | 114 |
| 6.5 | 圆角: border-radius | 116 |
| 6.6 | 投影 | 118 |
| 6.7 | 文本阴影 | 122 |
| 6.8 | 更多阴影 | 122 |

| | |
|--|-----|
| 6.9 小结 | 123 |
| 第7章 CSS3 渐变和多背景 | 125 |
| 7.1 线性渐变 | 126 |
| 7.1.1 W3C语法 | 127 |
| 7.1.2 旧WebKit语法 | 130 |
| 7.1.3 实际应用 | 131 |
| 7.1.4 使用SVG的线性渐变 | 133 |
| 7.1.5 使用Internet Explorer滤镜的线性渐变 | 135 |
| 7.1.6 便捷的工具 | 136 |
| 7.2 径向渐变 | 136 |
| 7.2.1 W3C语法 | 137 |
| 7.2.2 旧WebKit语法 | 139 |
| 7.2.3 实际应用 | 140 |
| 7.3 重复渐变 | 141 |
| 7.4 多背景图像 | 142 |
| 7.5 背景大小 | 145 |
| 7.6 小结 | 147 |
| 第8章 CSS3 转换和过渡 | 149 |
| 8.1 转换 | 149 |
| 8.1.1 平移 | 150 |
| 8.1.2 缩放 | 152 |
| 8.1.3 旋转 | 153 |
| 8.1.4 倾斜 | 154 |
| 8.1.5 更改转换的原点 | 154 |
| 8.1.6 对Internet Explorer 8及更早版本的支持 | 155 |
| 8.2 过渡 | 156 |
| 8.2.1 transition-property | 157 |
| 8.2.2 transition-duration | 158 |
| 8.2.3 transition-timing-function | 159 |
| 8.2.4 transition-delay | 160 |
| 8.2.5 transition简写属性 | 160 |
| 8.2.6 多个过渡 | 161 |
| 8.3 动画 | 162 |
| 8.3.1 关键帧 | 162 |

| | |
|------------------------------------|------------|
| 8.3.2 动画属性 | 164 |
| 8.4 小结 | 167 |
| 第9章 嵌入字体和多列布局 | 169 |
| 9.1 Web 字体和@font-face | 169 |
| 9.1.1 实现@font-face | 170 |
| 9.1.2 声明字体来源 | 172 |
| 9.1.3 字体属性描述符 | 174 |
| 9.1.4 Unicode范围 | 175 |
| 9.1.5 应用字体 | 176 |
| 9.1.6 法律因素 | 176 |
| 9.1.7 创建各种字体文件类型: Font Squirrel | 177 |
| 9.1.8 其他考虑因素 | 180 |
| 9.2 CSS3 多列布局 | 180 |
| 9.2.1 column-count属性 | 181 |
| 9.2.2 column-gap属性 | 182 |
| 9.2.3 column-width属性 | 182 |
| 9.2.4 columns简写属性 | 184 |
| 9.2.5 列和height属性 | 184 |
| 9.2.6 其他列功能 | 185 |
| 9.2.7 其他考虑因素 | 186 |
| 9.2.8 渐进增强 | 187 |
| 9.3 媒体查询 | 188 |
| 9.3.1 什么是媒体查询 | 188 |
| 9.3.2 语法 | 189 |
| 9.3.3 媒体查询的灵活性 | 190 |
| 9.3.4 浏览器支持 | 190 |
| 9.3.5 其他阅读材料 | 191 |
| 9.4 小结 | 191 |
| 第10章 地理定位、离线 Web 应用和 Web 存储 | 193 |
| 10.1 地理定位 | 194 |
| 10.1.1 隐私问题 | 195 |
| 10.1.2 地理定位方法 | 195 |
| 10.1.3 使用Modernizr检查支持 | 196 |
| 10.1.4 获取当前位置 | 196 |

| | | |
|---------|---------------------------|-----|
| 10.1.5 | 地理定位的Position对象 | 197 |
| 10.1.6 | 获取经度和纬度 | 198 |
| 10.1.7 | 加载地图 | 199 |
| 10.1.8 | 关于旧式移动设备的结束语 | 204 |
| 10.2 | 离线 Web 应用 | 204 |
| 10.2.1 | 工作原理: HTML5应用程序缓存 | 205 |
| 10.2.2 | 设置站点离线工作 | 205 |
| 10.2.3 | 获取离线存储站点的权限 | 208 |
| 10.2.4 | 离线测试 | 208 |
| 10.2.5 | 使The HTML5 Herald离线可用 | 210 |
| 10.2.6 | 离线Web应用存储的限制 | 211 |
| 10.2.7 | 后备部分 | 211 |
| 10.2.8 | 刷新缓存 | 213 |
| 10.2.9 | 我们在线吗 | 214 |
| 10.2.10 | 其他阅读材料 | 215 |
| 10.3 | Web 存储 | 215 |
| 10.3.1 | 两种存储 | 216 |
| 10.3.2 | Web存储数据的外观 | 217 |
| 10.3.3 | 获取和设置数据 | 218 |
| 10.3.4 | 转换存储的数据 | 218 |
| 10.3.5 | 快捷方式 | 219 |
| 10.3.6 | 删除条目和清除数据 | 219 |
| 10.3.7 | 存储限制 | 219 |
| 10.3.8 | 安全考虑 | 220 |
| 10.3.9 | 将Web存储添加到The HTML5 Herald | 221 |
| 10.3.10 | 用网页审查工具查看Web存储值 | 224 |
| 10.4 | 其他 HTML5 API | 226 |
| 10.4.1 | 网络工作者 | 226 |
| 10.4.2 | 网络套接字 | 227 |
| 10.4.3 | Web SQL和IndexedDB | 227 |
| 10.5 | 返回到绘制面板 | 228 |
| 第 11 章 | 画布、SVG 和拖放 | 229 |
| 11.1 | 画布 | 229 |
| 11.1.1 | 关于画布的一些历史 | 230 |
| 11.1.2 | 创建画布元素 | 230 |

| | | |
|---------|--------------------------|-----|
| 11.1.3 | 在画布上绘制 | 232 |
| 11.1.4 | 获取背景 | 232 |
| 11.1.5 | 用颜色填充画笔 | 233 |
| 11.1.6 | 在画布上绘制矩形 | 234 |
| 11.1.7 | 画布坐标系 | 234 |
| 11.1.8 | fillStyle的变化 | 235 |
| 11.1.9 | 通过创建路径绘制其他形状 | 237 |
| 11.1.10 | 存储画布绘制 | 240 |
| 11.1.11 | 在画布上绘制图像 | 241 |
| 11.1.12 | 处理图像 | 243 |
| 11.1.13 | 将彩色图像转换为黑白图像 | 244 |
| 11.1.14 | getImageData的安全性错误 | 247 |
| 11.1.15 | 用画布测试视频 | 247 |
| 11.1.16 | 在画布上显示文字 | 250 |
| 11.1.17 | 关注可访问性 | 254 |
| 11.1.18 | 其他阅读材料 | 254 |
| 11.2 | SVG | 254 |
| 11.2.1 | 在SVG上绘制 | 255 |
| 11.2.2 | 使用Inkscape创建SVG图像 | 258 |
| 11.2.3 | SVG过滤器 | 258 |
| 11.2.4 | 使用Raphaël库 | 259 |
| 11.2.5 | 画布与SVG | 261 |
| 11.3 | 拖放 | 262 |
| 11.3.1 | 给WAI-ARIA猫喂食 | 263 |
| 11.3.2 | 使元素可拖动 | 264 |
| 11.3.3 | DataTransfer对象 | 265 |
| 11.3.4 | 接受可以放下的元素 | 266 |
| 11.3.5 | 其他阅读材料 | 269 |
| 11.4 | 结束了，朋友们！ | 269 |
| 附录 A | Modernizr | 271 |
| 附录 B | WAI-ARIA | 277 |
| 附录 C | 微数据 | 281 |

HTML5 和 CSS3 简介

本章将简要回顾 HTML5 和 CSS3 的发展历程，并介绍 HTML5 和 CSS3 对于现代网站及 Web 应用程序的重要性，以及如何应用这些技术。

当然，如果您希望直接进入创建项目的实质部分，并开始学习如何使用 HTML5 和 CSS3 的新技术及功能，您可以先跳到第 2 章，稍后再回到本章。

1.1 什么是 HTML5

我们今天所理解的 HTML5，它具有一段相对动荡的历史。您可能已经了解到 HTML 是万维网上用于描述网页内容及数据的主要标记语言。HTML5 是此标记语言的最新版本，它包括新功能、对现有功能的改进以及基于脚本的 API。

也就是说，HTML5 兼容以前的所有版本——包括 HTML4 和 XHTML1.0 的所有有效元素。此外，在设计它时考虑到了一些主要原则，以确保在每个平台上能够正常工作，兼容所有的旧版浏览器并恰当地处理错误。您可在 W3G's HTML Design Principles 网页¹查阅创建 HTML5 的设计原则概述。

首先，HTML5 包括了现有标记元素的重新定义以及可使网络设计人员在标记

¹ <http://www.w3.org/TR/html-design-principles/>

语义时更具表现力的新元素。在您可以使用 `articles`、`sections`、`headers` 和 `footers` 等元素时，为什么还要用 `div` 元素将您的网页弄的看起来很杂乱？

“HTML5”另外还用于特指许多其他的新技术及 API。其中一些包括用 `<canvas>` 元素绘制、离线存储、新的 `<video>` 和 `<audio>` 元素、拖放功能、微数据和嵌入字体等。本书将涵盖许多诸如此类的新技术。



什么是 API？

API 表示应用程序编程接口。用理解图形用户界面（GUI）的思维方式去理解 API——除了是用户接口，也是代码接口。API 为您的程序提供一组“按钮”（预定义方法），通过单击，可以从系统、软件库或浏览器进行相关操作。

基于 API 的命令是将在后台完成（或有时由第三方软件完成）的更复杂的内容进行抽象处理的一种方式。一些与 HTML5 相关的 API 将在本书后面的章节中进行介绍和讨论。

总之，如果没有 JavaScript 或其他基于脚本的 API 相关经验，您也不必感到担心。尽管具有相关经验，肯定有益处，但这并不是必不可少的。

不管怎样，我们将循序渐进地带您浏览本书的脚本部分，以确保您能够完全掌握！

另外，应该注意在本书中，以前是 HTML5 一部分的一些技术，已经从本规范中分离开来，从技术角度讲，他们已不属于“HTML5”范畴。一些确定不再是 HTML5 一部分的技术，有时也被列在此类下。所以我们使用广泛的、包含一切的表达方式，比如“HTML5 及相关技术”。Bruce Lawson 甚至半开玩笑地建议使用 NEWT（New Exciting Web Technologies，令人兴奋的新 Web 技术）¹ 术语来表述。

当然，为了方便起见（并且避免引起争论），我们将这些技术统称为“HTML5”。

1.2 如何发展至今

Web 设计产业发展历史相对较短。12 年前，包含图像和醒目的设计，就被认为是网页内容的“最高级别”。

¹ <http://www.brucelawson.co.uk/2010/meet-newt-new-exciting-web-technologies/>

现在，情形大不一样。简单、性能驱动、重要功能依赖于客户端脚本的基于 Ajax 的 Web 应用程序越来越普遍。如今的网站经常类似于独立计算机软件应用程序，并且越来越多的开发人员关注它。

与此同时，Web 标记也随之发展。HTML4 最终给 XHTML 让步。XHTML 其实就是具有严格 XML 样式语法的 HTML4。目前，普遍应用 HTML4 和 XHTML，但是 HTML5 正在飞速发展。

HTML5 最初始于两种不同的规范：Web Form 2.0 和 Web Apps 1.0。两者都是改变网页外观，以及满足更快、更高效和易于维护的 Web 应用程序需求的产物。表单与应用程序类功能是 Web 应用程序的核心，所以这自然是 HTML5 规范的方向。最后，这两种规范合二为一，形成了今天的 HTML5。

在 HTML5 不断发展期间，XHTML 2.0 也在发展。这个项目曾经被搁置，以便于集中研究 HTML5。

实际 HTML5 规范便于使用吗

由于 HTML5 规范是由两部分发展而成的（WHATWG 和 W3C），所以形成了两种不同规范版本。W3C（或 World Wide Web Consortium）您可能比较熟悉：这是一个组织，它保持了原始的 HTML 和 CSS 规范，并且用于其他相关标准，比如 SVG（可缩放矢量图形）和 WCAG（Web Content Accessibility Guidelines）。

WHATWG（Web Hypertext Application Technology Working Group）可能对您来说比较陌生。一群来自于 Apple、Firefox 及 Opera 的人员，在 2004 年的 W3C 会后成立了该组织。当时他们认为 W3C 忽略了浏览器制造商和用户的需求以及 HTML 标准的向后兼容，而是专注于 XHTML 2.0 研究。所以他们独自开发了上面提到的 Web 应用程序和 Web 表单规范，随后这两个规范进行了合并，新规范被称为 HTML5。看到这些，W3C 最后妥协并根据 WHATWG 的规范创立了他们自己的 HTML5 规范。

这些可能让人们感到有些糊涂。是的，在这背后有一些政治因素。我们作为设计人员和开发人员，并不能控制这些。但这产生了两个版本的规范，是否会困扰我们？简单地讲，不会。

WHATWG 的规范版本可以在 <http://www.whatwg.org/html> 上找到，最近将此版本重命名为“HTML”（去掉了“5”）。现在它被称为“活标准”，意指此标准将会不断发展，并不再使用增量版本号进行标注¹。

¹ 参见 <http://blog.whatwg.org/html-is-the-new-html5/> 可了解关于此更改的说明

WHATWG 版本包含的信息仅涵盖 HTML 功能，包括 HTML5 的新功能。另外，还另有一些由 WHATWG 开发的规范，涵盖了相关技术。这些规范包括微数据、二维环境画布、网络工作者和 Web 存储等¹。

W3C 的版本可以在 <http://dev.w3.org/html5/spec/> 找到，一些关于其他技术的独立规范可在 <http://dev.w3.org/html5> 上找到。

那么 W3C 与 WHATWG 两种版本的规范有什么区别呢？简单地讲，WHATWG 版本稍显不正规并具有一定的实验性（有一些可能有争议，思想更超前）。但总体上讲，他们非常相似，所以任何一个版本都可用作学习 HTML5 新元素和相关技术的基础。

1.3 我们为什么应关注 HTML5

如前所述，HTML5 的核心部分是新语义元素，以及相关技术和 API。通过介绍这些新内容和语言更改，目的是使网页更容易编写、使用和访问。

这些新语义元素，以及类似于 WAI-ARIA 和微数据（分别在附录 A、附录 B 和附录 C 中介绍）等其他标准，可以帮助使用人员及计算机更轻松地访问文档，从而有利于可访问性和搜索引擎的优化。

具体地讲，新语义元素适用于设计动态网页，使制作的网页更具模块性和可移植性。我们将在后面的章节中详细介绍。

最后，与 HTML5 相关的 API 可以帮助改进网络开发人员多年使用的技术。一些普通任务更简化了，使开发人员的能力变得更强大。此外，通过介绍基于 HTML5 的音频和视频元素，这意味着在网站上发布丰富的多媒体内容时，将会减少对第三方软件及插件的依赖性。

总的来说，这些都是我们学习 HTML5 新功能和 API 的原因所在。我们将在本书中对原因进行详细介绍。

1.4 什么是 CSS3

创建网页的另一个独立的但并非不重要的一部分是层叠样式表（CSS）。正如您所了解到的，CSS 是一种样式语言，用来描述如何呈现或设置 HTML 标记样式。

¹ 有关详细信息，请参见 http://wiki.whatwg.org/wiki/FAQ#What_are_the_various_versions_of_the_spec.3F

CSS3 是 CSS 规范的最新版本。术语“CSS3”不仅是 CSS 新功能的参考文献，也是 CSS 规范发展进程的第三级¹。

CSS3 包含 CSS2.1（CSS 规范的前一版本）的所有内容。它还添加了一些新功能来帮助开发人员解决一些问题，并不再需要非语义标记、复杂的脚本及其他图像。CSS3 的新功能包括支持附加选择符、投影、圆角、多背景、动画和透明度等。

CSS3 与 HTML5 是截然不同的。在这本书中，我们将用术语“CSS3”来特指 CSS 规范的第三级，并集中介绍其新功能。这样，CSS3 独立于 HTML5 及其相关的 API。

1.5 我们为什么应关注 CSS3

在这本书的后面，我们将详细介绍 CSS3 的新功能。同时，我们将告诉您为什么 CSS3 的新技巧能够令网络开发人员兴奋。

一些设计技巧几乎在每一个项目中都会得到应用。投影、渐变和圆角是 3 个非常好的示例。它们应用在每一个地方。只要应用恰当，并与网站的整体主题和目的一致，这些技术的改进会为整个设计增添光彩。

可能您在想：我们使用 CSS 创建这些设计元素已经很多年了。我们还有必要学这些吗？

过去，为了创建渐变、投影及圆角，网络设计人员必须求助于许多棘手的技巧。有时，还需要一些其他的 HTML 元素。在某些情况下，HTML 保持的相当整洁，但此时需要一些脚本使用技巧。以渐变为例，使用其他图标是不可避免的。我们忍受这种解决方法，是因为我们没有其他方法完成这些设计。

CSS3 允许我们以这种思想超前的方式使用这些和其他的设计元素，从而使我们在诸多方面受益：使标记整洁；从而便于使用人员及计算机进行访问；维护代码；减少不必要的图标以及更快载入网页。



供应商前缀注释

为了使用 CSS3 的一些新功能，需要包含几行特殊的代码。这是因为浏览器供应商在添加执行 CSS3 的一些新功能时，使用了自己的前缀版本属性。

¹ <http://www.w3.org/Style/CSS/current-work.en.html>

例如，在 FireFox 中转换元素时，需要使用 `-moz-transform` 属性；同样，在基于 WebKit 的浏览器中，比如，Safari 和 Google Chrome，您必须使用 `-webkit-transform` 属性。在有些情况中，为使用一个 CSS 属性，您可能必须添加 4 行代码。这可能似乎会丧失一些从避免黑客、图像及非语义标记所获得的益处。

但是，浏览器供应商以这种方式执行这些新功能，理由如下：现在的规范还不是最终版本，在程序执行中，还有一些漏洞。因此，现在您在执行这些功能时，使用供应商前缀来提供数值，并且使用无前缀声明来提供每个属性的永久版本。当规范成为最终版本且实现经过完善后，浏览器前缀将被取消。

即使用这些前缀来维护代码似乎需要很多工作，现在使用 CSS3，仍然是利大于弊。除了需要改变一些前缀属性来修改设计元素外，维护基于 CSS3 的设计还是比通过图形程序更改背景图像或处理那些其他标记和黑客脚本所带来的弊端要容易一些。此外，如上所述，您的代码更不会过时。

1.6 在现实中，我们想做的是什

在现实中，我们不会创建了一个网站后，然后去做另一个项目，而对前一个项目的工作置之不理。我们会创建 Web 应用程序，并对其进行更新、微调，对其潜在的性能问题进行测试，调整其设计、布局和内容。

换句话说，在现实中，我们都会再次访问我们所写的代码。所以我们编写代码时，会使用最可靠、易于维护并高效的方法，并便于以后访问代码对其进行必要的完善和修改。显然，这不仅对我们自己创建和维护的网站及 Web 应用程序十分重要，同样，也对为我们客户所创建和维护的网站及 Web 应用程序同等重要。

我们需要不断地探讨新方法和更好的方式编写我们的编码。HTML5 和 CSS3 在此方面迈出了一大步。

1.6.1 多种浏览器的市场

尽管 HTML5 还在发展中，并在标记内容方式上呈现了巨大的变化，但是值得注意的是，这些变化并没有引起旧版浏览器不兼容，或版面问题及页面错误。

这就意味着您可以在您的项目中使用任何有效的 HTML4 或 XHTML 标记，将 `doctype` 更改为 HTML5（将在第 2 章中介绍），并且网页仍旧有效并与之前的显示相

同。在 HTML5 中新添加和改进的标记功能可以这种方式实施在编程语言中，以确保向后兼容老的浏览器（甚至是 IE6）！

但那些仅是标记，那么其他的 HTML5 新功能、CSS3 以及相关技术呢？根据一组数据表明¹，大约 47% 的用户所用的浏览器版本不支持大部分新功能。

因此，开发人员便想出各种解决方案向那些采用 HTML5 和 CSS3 新功能的用户提供同等经验。有时，就像提供一个备选方案一样简单。例如，向没有本机视频支持的浏览器提供一个 Flash 视频播放器。在其他情况下，也有必要使用脚本模拟支持新功能。这些“间隙填充”技巧称为 **polyfills**。在创建高性能 Web 应用程序时，依靠脚本模仿本机功能并不总是最好的方法，但为了力求发展使用这些新功能，有必要采用这种方法，尽管会增加一些痛苦。稍后我们会在本书中举例论证。

所以，我们向您推荐撤销选项及 polyfill 以解决浏览器的不兼容性。我们将尽力提醒您使用这些选项时存在的潜在弊端和隐患。

当然，有时候也根本不需要撤销选项及 polyfill。例如，当我们使用 CSS3 在框中创建圆角时，在老的浏览器中会看到正方形框，但没有任何影响。网站的功能性并没有降级，那些用户也没有任何损失。

关于对浏览器的有限支持，您可能感觉有些气馁。不要这样！这里有个好消息，全世界超过 40% 的用户所使用的浏览器都对我们在本书中讨论的新内容提供支持。这种支持每一刻都在增长，新浏览器版本（比如 Internet Explorer 9）继续添加对这些新功能和技术的支

持。我们在发展过程中不断吸取教训，因此我们将确切地通知您哪些地方还缺少支持，以便于您在使用 HTML5 和 CSS3 时，知道所创建的哪些内容可以呈现在读者面前。即使没有 HTML5 和 CSS3 的新功能，我们也将讨论一些方法，确保您在使用没有获得技术支持的浏览器时得到一些可行性方案。

1.6.2 不断发展的移动市场

如今开始学习并使用 HTML5 和 CSS3 的另一个原因是迅速发展的移动市场。

根据 2009 年的市场调研机构 StatCounter 的报告，仅有超过 1% 的用户在手机上使用 Internet²。在不到两年的时间里，这个数字以 4 倍的速度增长，现已超过 4%³。

¹ http://gs.statcounter.com/#browser_version-ww-monthly-201011-201101-bar

² http://gs.statcounter.com/#mobile_vs_desktop-ww-monthly-200901-200912-bar

³ http://gs.statcounter.com/#mobile_vs_desktop-ww-monthly-201011-201101-bar

一些报告显示,根据不同的分析方法,这些数据值可能更高。无论如何,移动市场在以惊人的速度增长,这是一个不争的事实。

4%的使用率似乎看起来很小,公正地说,确实是这样。但是它的增长率是引人注目的——两年内增长 400%。对于那些学习 HTML5 和 CSS3 的人来说,这意味着什么呢?

HTML5、CSS3 以及相关的前沿技术在许多移动 Web 浏览器中得到了很好的支持。例如,iOS 设备(比如 iPhone 和 iPad)上的 Safari、Opera Mini 和 Opera Mobile,以及 Android 操作系统的 Web 浏览器都对 HTML5 和 CSS3 提供了强大的支持。由这些浏览器所支持的新功能及技术包括 CSS3 的颜色和不透明度、画布 API、Web 存储、可缩放矢量图形、CSS3 圆角和离线 Web 应用等。

实际上,我们在本书中将要介绍的一些新技术设计时已考虑了移动设备。已设计了离线 Web 应用和 Web 存储等技术,这在一定程度上是由于越来越多的人通过移动设备访问网页。此类设备对在线数据使用通常有一些限制,因此脱机访问 Web 应用程序会很有用处。

我们将在本书第 10 章介绍这些主题。当然,我们也将通过本书的其他章节向您介绍针对各种设备和平台创建网页所需的工具。

1.7 实际应用

推出新技术并希望此技术只为某一级别的浏览器提供编写网页和应用程序的服务,这显然是不现实的。在现实世界中,我们希望 HTML5 和 CSS3 取得进一步的发展,所以我们要使它们能够在各种平台上开发网页。这种平台包括现代浏览器、Internet Explorer 的早期版本以及正在蓬勃发展的移动设备市场。

在某些时候,可以为不同的用户代理提供不同的指令,但这类似于早期具有混乱的浏览器嗅探和代码分叉的 Web。但是这一次不一样,新代码是面向未来的,所以当旧版浏览器不能正常使用时,您仅需删除回调和 polyfill,将基本代码放在那里以供现代浏览器使用即可。

HTML5 和 CSS3 是引导我们进入编写网页精彩世界的先驱技术。因为所有的现代浏览器(包括 Internet Explorer 9)都对 HTML5 和 CSS3 的许多新功能提供强大的支持,所以与以往相比,开发人员能够更轻松地创建功能强大、易于维护及永不过时的网页。

由于旧版浏览器所占的市场份额逐渐减少，因此您今天所学习的 HTML5 和 CSS3 的技术将更具价值。通过学习这些技术，您在网页设计方面将有一个光明的未来。我们对为什么要学习 HTML5 和 CSS3 做了充足的解释，下面我们将开始深入介绍“如何学好它们”。

HTML5 样式的标记

我们已经向您介绍了关于 HTML5 历史的一些基本知识，以及学习它的重要性。现在，我们将在您的项目中开始使用它。首先向您介绍我们将在本书中逐步创建的示例网站。

在简单介绍了我们将创建的示例网站之后，我们将讨论一些 HTML5 的基本语法知识，以及编码最佳实践的一些建议。我们将遵循浏览器兼容性及 HTML5 页面结构基本原理等一些重要原则。最后，我们向您介绍一些 HTML5 特有的元素，以及如何使它们适应布局。

现在开始吧！

2.1 The HTML5 Herald 简介

我们首先从头开始创建一个示例网站项目。

该网站已经创建完毕，请参见 <http://thehtml5herald.com/>。这是一个老式的报纸风格的网站，名为 The Html5 Herald。网站的主页包含一些以视频、图片、文章和广告形式呈现的媒体，而且在另一个页面上包含一个注册表。

我们来看一下源代码，如果愿意可以尝试一些功能。在我们学习本书时，我们将通过编写代码来制作网站。我们不可能涉及有关 CSS 的每一个细节，因为您可能

已经熟悉了大部分内容：浮动布局、绝对和相对定位，以及基本字体样式等。我们将主要介绍 HTML5 的新增元素、API，以及所有用于为各种元素添加样式及互动的新的 CSS 技术。

图 2.1 所示为已创建网站的外观示意。



图 2.1 The HTML5 Herald 的主页

构建网站时，我们会详细解释 HTML5 的新增元素、API 以及 CSS3 的新功能，同时，我们也将向您推荐一些最佳实践。当然，其中的一些新技术还在开发中，所以我们不可能武断地告诉您这些技术可以做什么，以及不可以做什么。

2.2 基本的 HTML5 模板

在您学习 HTML5 和新技术时，您可能想创建自己的蓝图或示例文件，并通过它们来构建基于 HTML5 的项目。实际上，您可能对已存在的 XHTML 或 HTML4.0 项目已经做了类似的事情。我们鼓励您这样做，当然您也可以考虑使用一些在线资源，从而为您提供一个 HTML5 的基本起点¹。

在这个项目中，我们将从头开始，编写自己的代码，并对每一部分进行详细讲

¹ 想要了解更多内容，请访问 <http://www.html5boilerplate.com/> 和 <http://html5reset.org/>

解。当然，即使是最好、最庞大的示例网站也不可能包含所有的新元素和技术。我们也会详细讲解一些未在我们所创建的示例网站中应用的新功能。这样，在您决定如何创建 HTML5 和 CSS3 网站和 Web 应用程序时，就会熟悉所提供的各种选项，也就可以将本书当作许多技术的快速参考资料了。

让我们从一个简单的 HTML5 网页的基本框架开始：

index.html (excerpt)

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">

  <title>The HTML5 Herald</title>
  <meta name="description" content="The HTML5 Herald">
  <meta name="author" content="SitePoint">

  <link rel="stylesheet" href="css/styles.css?v=1.0">

  <!--[if lt IE 9]>
    <script src="http://html5shiv.googlecode.com/svn/trunk/html5.js">
  </script>
  <![endif]-->
</head>
<body>

  <script src="js/scripts.js"></script>
</body>
</html>
```

仔细看一下上面的标记。如果您正在从 XHTML 或 HTML4 过渡到 HTML5，您会立即注意到 HTML5 在许多方面是不同的。

2.2.1 Doctype

首先，我们进行文档类型声明（Document Type Declaration，也称为 doctype）。它可以用来告诉浏览器（或任何其他分析程序）它们所查看的文件类型。在 HTML 文件中，它表示具体的 HTML 版本及风格。doctype 应是位于 HTML 文件最顶端的第一个项目。过去，doctype 声明非常难看且很难记，对于 XHTML 1.0 严格类型来说，doctype 声明如下所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

对于 HTML4 过渡类型来说, doctype 声明如下所示。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

经过几年的发展, 代码编辑软件开始提供包含 doctype 的 HTML 模板, 或提供自动插入模板的方式。当然, 快速的网页搜索会很轻松地调出代码以插入您需要的文档类型。

尽管在文档顶端的一长串文本并没有对我们造成太大的影响(没有迫使我们的网站浏览者下载其他的字节), HTML5 清理了这个难以辨认的眼中钉。现在, 我们所需要的只是下面这一行代码。

```
<!doctype html>
```

既简单又明了。请您注意, “5” 已经在声明里消失了。尽管目前网站标记的版本是“HTML5”, 但这确实仅是之前的 HTML 标准的更新——以后的规范还会在今天的基础上进一步发展。由于浏览器要支持网站的所有内容, 因此这里没有一个固定的文档类型去告诉浏览器应支持文档中的哪种功能。

2.2.2 html 元素

任何 HTML 文件中所包含的内容都是 html 元素, 该元素在 HTML5 中并没有显著的变化。在我们的示例中, 包含了值为“en”的 lang 属性, 表示文档的语言是英语。在基于 XHTML 的语法中, 要求包含 xmlns 属性。在 HTML5 中, 已经没有这项要求。即使是 lang 属性, 对于文档的验证或正确运行都不是必不可少的。

我们目前所拥有的标记如下所示, 其中包括结束</html>标记:

```
<!doctype html>
<html lang="en">

</html>
```

2.2.3 head 元素

页面的下一部分是<head>部分。head 中的第一行用来定义文档的字符编码。这是另一个被简化的元素。它的用法如下所示。

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

HTML5 通过将字符编码<meta>标签的内容缩减到最少，从而使之得到改进。代码如下所示。

```
<meta charset="utf-8">
```

在几乎所有的情况下，utf-8 将是您在文档中使用的值。本章篇幅有限，这里将不能完整地介绍字符编码，或许您对此也并不感兴趣。当然，如果您想进一步研究，可以访问 W3C 网站¹查阅相关主题。



预先准备

为确保所有浏览器能够正确读取字符编码，整个字符编码声明必须包含在文档的前 512 个字符中。必须将它放在所有基于内容的元素前面（比如，<title>元素，我们将在示例网站中演示）。

关于这个主题，我们可以写很多。但是，为了使你们节省精力，我们省去了一些细节。现在，我们可以满意地接受这个简化的声明，并继续我们文档的下一部分。

```
<title>The HTML5 Herald</title>
<meta name="description" content="The HTML5 Herald">
<meta name="author" content="SitePoint">

<link rel="stylesheet" href="css/styles.css?v=1.0">
```

在这几行中，HTML5 与以前版本的语法几乎没有什么不同。页面标题的声明和以前一样，<meta>标签仅是一个可选的示例，用来表示可放置的位置，您可以按照自己的意愿，在此放置任意多个 meta 元素。

此标记块的主要部分是样式表，我们使用习惯的 link 元素来包含它。初看起来，您可能没有发现任何区别。但是按照惯例，link 元素包含一个 type 属性，其值为 text/css。有趣的是，在 XHTML 或 HTML4 中不需要该属性，即使是在使用严格的文档类型时也是如此。因为所有的浏览器都可以识别链接样式表的

¹ <http://www.w3.org/TR/html-markup/syntax.html#character-encoding>

内容类型，而无须其他的属性，所以，基于 HTML5 的语法鼓励您完全放弃使用 type 属性。

2.2.4 公平竞争

接下来，在介绍标记中的新元素之前，我们需要先了解一些背景知识。

HTML5 包括一些新元素，比如，我们将稍后介绍的 article 和 section 元素。您可能认为是旧版浏览器的主要问题，但这是错误的想法。这是因为大多数浏览器实际上并不在乎您使用什么标签。如果您的 HTML 文档有 <recipe> 标签（或甚至使用 <ziggy> 标签），或者 CSS 将一些样式附加到该元素上，几乎所有浏览器都会正常运行，并应用该样式。

当然，这个假设的文档不会生效，但在大多数浏览器（Internet Explorer 是一个例外）上可正确呈现。在 Internet Explorer 9 之前，Internet Explorer 禁止接收无法识别的元素。呈现引擎视这些秘密元素为“未知元素”，所以您不能改变其外观及行为方式。这不仅包括我们所设想的元素，还包括那些在所开发的浏览器中未定义的元素。当然也包括 HTML5 的新元素。

编写本书时，Internet Explorer 9 刚刚发布（被广泛使用还会需要一段时间），所以这是一个问题。我们想开始使用这些光鲜的新标签，但是如果不能将这些 CSS 规则附加到这些元素上，那么我们的设计就会崩溃。

幸运的是，这里有一个解决方案：一个非常简单的 JavaScript，最初由 John Resig 开发，可以神奇地使 HTML5 的新元素在 Internet Explorer 的早期版本中显示。

我们已经在条件注释的标签 <script> 中包含了所谓的“HTML5 shiv”¹。条件注释是 Microsoft 在 Internet Explorer 中实施的一个专用功能。它们向您提供用脚本或样式确定特定浏览器版本的能力²。这种条件注释告诉浏览器：封装的标记仅供用户使用 Internet Explorer 9 之前的版本查看网页。

```
<!--[if lt IE 9]>
<script src="http://html5shiv.googlecode.com/svn/trunk/html5.js">
</script>
<![endif]-->
```

¹ 您可能很熟悉它的另一个名称，即 HTML5 shim。它们具有相同的代码片段，我们将所有实例都称为 HTML5 shiv（原始名称）

² 有关更多信息，请参见 <http://reference.sitepoint.com/css/conditionalcomments>

请注意,如果您用 JavaScript 库处理 HTML5 的新功能或新 API,可能已经使 HTML5 的脚本可以运行。在本例中,您可以删除关于 Remy Sharp 的脚本。其中的一个示例是 Modernizr¹,这是一个 JavaScript 库,它能检测现代 HTML 和 CSS 的新功能,将在附录 A 介绍它。Modernizr 包括能够使 HTML5 在 Internet Explorer 早期版本中显示的代码,所以 Remy 脚本就显得多余了。



使用 Internet Explorer 6~Internet Explorer 8 但禁用了 JavaScript 的用户该怎么办呢?

当然,仍有一些用户(禁用了 JavaScript 的用户)不能够使用 Remy 的 HTML5 shiv,可能有这个或那个原因。作为网站设计人员,即使用户不能使用 JavaScript,我们也必须使所有用户都能够浏览我们创建的网站内容。当 45%~75%的用户使用 Internet Explorer 时,这似乎是一个严重的问题。

但是事实并没有那么糟糕。许多研究表明,不能使用 JavaScript 的用户非常少,可以忽略不计。

2010 年 10 月在雅虎网站发布的一项研究²表明,全世界不能使用 JavaScript 的用户大约只有 1%。而另一个对数十亿用户的研究³也得到了几乎相同的结果。在这两项研究中,相对于世界其他地方,美国用户不能够使用 JavaScript 的人数最多。

还有其他一些使用 HTML5 新元素的方式,不需要 JavaScript 在不受支持的浏览器中显示设置了样式的元素。不幸的是,那些方法很不现实,有许多其他缺陷。

如果您仍然十分关心这些用户,可以考虑这种混合方法。例如,在缺少样式的地方使用新 HTML5 元素不会有大问题,同时对于主要的布局容器,可使用 div 等传统的元素。

2.2.5 剩余部分是历史简介

让我们看看启动模板的剩余部分,我们通常使用 body 元素以及其结束标记和 </html> 的结束标记。我们还在 script 元素中包含 JavaScript 文件的引用。

与我们前面讨论过的 link 元素非常相像,<script>标签并不要求您声明 type

¹ <http://www.modernizr.com/>

² <http://developer.yahoo.com/blogs/ymn/posts/2010/10/how-many-users-have-javascript-disabled/>

³ <http://visualrevenue.com/blog/2007/08/eu-and-us-javascript-disabled-index.html>

属性。在 XHTML 中验证包含外部脚本的网页，<script>标签应如下所示：

```
<script src="js/scripts.js" type="text/javascript"></script>
```

出于实用性目的，JavaScript 仅是一种在 Web 上使用的脚本语言，并且即使您不明确声明，所有的浏览器也都会假设您使用的是 JavaScript，所以 type 属性无需在 HTML5 文档中声明：

```
<script src="js/scripts.js"></script>
```

我们将 script 元素放在页面底部，以使其符合嵌入 JavaScript 的最佳做法。我们还需处理页面的载入速度。当浏览器碰到脚本时，它会将脚本分解成若干部分，此时将暂停载入和呈现页面的其余部分。如果在页面顶部以及所有内容之前放置大量脚本，这将导致页面载入十分缓慢。这就是为什么将大部分脚本放在页面最底部的原因，这样可以在页面载入完成以后，再对脚本进行分解。

在一些情况（比如 HTML shiv）下，由于您想在浏览器开始呈现页面之前呈现效果，可根据需要，将脚本放在文档的 head 中。

2.3 HTML5 常见问题

在简要介绍了 HTML5 标记以后，您可能会有一些问题。以下是一些常见问题的答案。

2.3.1 为什么这些变更仍能够在旧版浏览器中工作

这是许多开发人员难于接受 HTML5 的地方。其实这根本不是问题，为了便于读者理解，我们会将 HTML5 和 CSS3 的一些新功能（后面章节将介绍）进行对比。

在 CSS 中，在添加新功能时（例如，border-radius 属性将圆角添加到元素中），同时也需要将它添加到浏览器呈现引擎中，因此旧版浏览器无法识别它。所以，用户在使用不支持 border-radius 的浏览器浏览网页时，圆角就变成了方形。CSS3 的其他功能也与其类似，所以在某种程度上降低了它的使用性能。

许多开发人员认为 HTML5 与 CSS 的工作方式类似。某些高级功能及我们在本书中稍后介绍的 API 可能是这样的，但并不是所有的更改都是如此，比如更简单的语法、减少的冗余和新文档类型。

在精心研究了旧版浏览器可以处理及不可以处理什么之后，才对 HTML5 语法进行了定义。例如，在 HTML5 中，至少需要 15 个字符组成文档类型声明，以便所有浏览器以标准模式显示页面。

同样，XHTML 需要在 html 元素中包含更长的字符编码声明和其他的属性来用于验证，浏览器并不需要他们来正确显示页面。再次声明，旧版浏览器经过了精心检测，最后决定简化字符编码并删除 xmlns 属性，浏览器仍将正确显示页面。

经过简化的 script 和 link 元素也属于“简化但不破坏旧页面”这一类别。我们上面看到的 Boolean 属性同样也属于此类别，浏览器通常忽略像 checked 和 disabled 这样的属性值，那为什么还要提供这些属性呢？

如第 1 章所述，您不应害怕使用 HTML5。设计该语言时，已经考虑到向后兼容性的问题，并以尽可能支持现有内容为目标。

与 CSS3 和 JavaScript 中的更改不同，必须在浏览器制造商确实对此进行支持以后，新添加部分才可得到执行。使用 HTML5 语法则不需要等待新版浏览器的发行。此外，当使用新的语义元素时，只需少量 JavaScript 代码即可与旧版浏览器兼容。



什么是标准模式？

当基于标准的 Web 设计处于起步阶段时，浏览器制造商们就面临着一个问题：支持新标准，在许多情况下，将会打破为旧版非标准浏览器设计的现有网页的向后兼容性。浏览器制造商需要一个信号来指示既定网页能够按照标准来呈现。他们在文档类型中发现这样一个信号：新的、符合标准的网页包含格式正确的文档类型，旧版非标准网页通常则没有。

将文档类型作为信号使用，浏览器可以在标准模式（它们尝试按照标准以呈现字母的方式呈现元素）和 quirks 模式（它们试图模仿旧版浏览器“离奇”的呈现能力，以确保页面按照意图呈现）之间进行切换。

可以肯定地说，在目前的开发领域，几乎每一个网页都有一个适当的文档类型，因此将以标准模式呈现；因此，您将不太可能要处理一个由 quirks 模式呈现的页面。当然，如果一个用户使用非常古老的浏览器（比如 Internet Explorer 4）浏览网页，将会使用该浏览器的呈现模式显示页面。这就是 quirks 模式所要模仿的，而且无论是否使用文档类型，它都将这样运行。

尽管 XHTML 和旧版的 HTML 文档类型包含了所引用规范的确切版本信息，但是浏览器并没有真正利用这种信息。只要呈现了一个看似正确的文档类型，它

们就将以标准模式呈现页面。因此，HTML5 的文档类型已经精简到了启动任何浏览器标准模式所需的最低限度。

有关更多信息及一张概述了不同浏览器以 quirks 模式呈现的原因的图表，请参见 Wikipedia¹。您也可以在 SitePoint 的 CSS 参考资料²中阅读关于标准模式和 quirks 模式的概述。

2.3.2 是否需要关闭所有标签

在基于 XHTML 的语法中，需要关闭所有元素，即使已经有了相应的结束标记（比如</html>）或在 void 元素的情况下，在标签结尾处有一个正斜杠。后者是不包含子元素（比如 input、img 或 link）的元素。

您仍可以在 HTML5 中使用该语法样式，您更喜欢它可能是由于为了保持编码的一致性和可维护性，但已不需要添加尾部反斜杠以用于 void 元素验证。继续我们的“精简”主题，HTML5 允许您在此类元素后面删除尾部反斜杠，这样您的标记看起来更加干净、整洁。

值得一提的是，在 HTML5 中，大多数元素可以包含嵌套元素，但如果恰巧是空的，仍然需要有一个相应的结束标记配对。此规则虽有例外，但是可以简单地认为这就是它的普遍性。

2.3.3 关于其他的基于 XHTML 的语法使用习惯

关于这个问题，其实省略结尾斜线仅是基于 HTML5 语法与 XHTML 不同的一方面。实际上，HTML5 验证完全忽略了语法风格问题，它只在编码错误以某种方式终止文件运行时才报错。

这就意味着如果能够通过验证，那么下面 5 行代码就是完全等同的：

```
<link rel="stylesheet" href="css/styles.css" />
<link rel="stylesheet" href="css/styles.css">
<LINK REL="stylesheet" HREF="css/styles.css">
<Link Rel="stylesheet" Href="css/styles.css">
<link rel=stylesheet href=css/styles.css>
```

在 HTML5 中，您可以使用小写、大写或大小写混合的标记名称或属性，同样

¹ http://en.wikipedia.org/wiki/Quirks_mode/

² <http://reference.sitepoint.com/css/doctypesniffing/>

也可以使用引用或未引用的属性值（只要这些值不包含空格或其他保留字符），都能通过验证。

在 XHTML 中，所有属性都必须有值，即使它们是多余的。例如，您经常会看到类似下面的标记：

```
<input type="text" disabled="disabled" />
```

在 HTML5 中，属性是“开”或“关”（称为布尔属性）可以不用指定任何值。因此上述 input 元素可以写成下列代码：

```
<input type="text" disabled>
```

因此，至少就语法而言，HTML5 的验证要求宽松了许多。这是否就意味着您可以无所顾忌并在任意指定元素上使用任何语法呢？不，我们当然不建议这样做。

我们鼓励开发人员选择一种语法风格，并坚持下去——特别是如果您在一个团队中工作，代码维护和可读性都是非常重要的。我们还建议（虽然这不是必须的）您选择最简单的编码风格，并保持一致。

以下是一些准则，您可以考虑使用。

- ❶ 所有元素和属性都小写，就像您使用 XHTML 一样。
- ❷ 除去那些不要求用结束标记的元素，我们建议关闭所有包含内容的元素（比如<p>Text</p>）。
- ❸ 尽管您可以不给属性值添加引号，但是极有可能您所编写的属性要求添加引号（例如，当声明由空格分开的多个类时，或在 URL 上附加查询字符串值时）。因此，为了保持统一性，我们建议您使用引号。
- ❹ 元素没有任何内容，可删除尾部反斜杠（比如 meta 或 input）。
- ❺ 避免为布尔属性提供多余值（例如，使用<input type="checkbox" checked>，而不使用<input type="checkbox" checked="checked">）。

并不是要求大家全部接受以上建议，但我们相信这些合理的语法建议能够实现干净标记，并易于阅读和维护。

如果您滥用编码风格，包含大量没有必要的东西，就是对 HTML5 创始者在力

求语言简化所取得的进展方面加以否定。

2.4 定义页面结构

现在我们已经了解了模板的基础知识，下面将开始介绍详细信息，为网页指定一些基本结构。

在本书后面的章节中，我们将深入介绍添加 CSS3 功能和其他 HTML5 的精华。现在，我们将考虑在网站总体布局方面应使用哪些元素。我们将在本节以及后面的章节介绍语义。当使用这个术语时，我们指的是用指定的 HTML 元素名称来描述其内容的方式。由于 HTML5 包含一系列广泛的语义元素，因此您会发现，与过去使用 HTML4 或 XHTML 的时候相比，多花了一些时间思考内容结构及含义。非常好！理解了内容的意思才能够写出好的标记。

如果您回头看一下 The HTML5 Herald 的截图（或登录在线网站），您会看到它被划分为以下几部分。

- 具有徽标和标题的标头部分。
- 导航栏。
- 主体内容分为 3 列。
- 列中的文章和广告块。
- 页脚包含一些作者和版权的信息。

在决定页面的不同部分采用哪些适当的元素之前，让我们考虑一些可选项。首先，我们将向您介绍一些新的 HTML5 的语义元素，这些元素可以帮助我们划分页面并为我们的文档结构增添更多的意义。

2.4.1 header 元素

很明显，我们要看的第一个元素是 header 元素。WHATWG 规范将它简单地描述为“一组前言或导航助手。”¹从本质上讲，这意味着，无论您习惯了在<div id="header">中包含什么内容，现在您都需要将这些内容放在元素 header 中。

¹ <http://www.whatwg.org/specs/web-apps/current-work/multipage/sections.html#the-header-element>

但是，这里有一个问题，header 元素与经常习惯用于网站标题部分的 div 元素有所不同：这里没有每页只用一次的限制。相反，您可以使用新的 header 元素介绍每节内容。我们在这里使用 section，实际上并不是下面将要介绍的 section 元素；从技术上讲，它是 HTML5 中的“分节内容”。在下一章中，我将详细地介绍这一内容。现在，您可以放心地将它理解为任意一段需要有自己的标头的内容。

header 元素可用于页面内任何特定独立部分的前言或导航助手，或应用于整个页面——或两者兼而有之。

虽然 header 元素经常会被放在页面或一节内容的顶部，但是它的定义与其位置无关。网站的布局可能要求将文章或博客的标题位于内容的左侧、右侧或下面。无论怎样，您仍可使用 header 元素来描述该内容。

2.4.2 section 元素

下一个元素是您所熟悉的 HTML5 的 section 元素。WHATWG 规范将其定义为：¹

section 元素表示文档或应用程序的通用部分。在此上下文中，一节就是一组专题内容，并通常带有标题。

它进一步解释为，section 元素不应仅用于以样式或脚本为目的的通用容器。如果您不能将 section 元素当作通用容器（例如，为了实现您想要的 CSS 布局），那么您该使用什么呢？我们的老朋友，div 元素，在语义上毫无意义。

让我们回到规范中的定义，section 元素的内容应是“主题的”，因此，以通用的方式使用它以包含不相关的内容是不正确的。

以下是一些恰当使用 section 元素的示例。

- 选项卡界面的各个部分。
- “关于”页面的部分，例如，一个公司的“关于”页面，可能包括公司的历史、使命及其团队等部分。
- 很长的“服务条款”页面的不同部分。

¹ <http://www.whatwg.org/specs/web-apps/current-work/multipage/sections.html#the-section-element>

在线新闻网站的各个部分。例如，文章可以分为体育、世界事务以及经济咨询等栏目。



语义

每一次向网络设计人员提供新的语义标记时，总会就如何正确使用这些元素进行讨论，比如规范的意图到底是什么等问题。您可能还记得关于在前一个 HTML 版本中如何正确使用 `dl` 元素的讨论。HTML5 也无法避免，尤其是涉及 `section` 元素时。

即使 Bruce Lawson，备受尊重的 HTML5 的权威，也承认曾在过去使用 `section` 元素不当。为了明确这一点，Lawson 发表¹了他对这些错误的解释，这是非常值得一看的。简单地讲：

- ❖ `section` 元素是通用的，所以如果有一个更具体的语义元素更恰当（比如 `article`、`aside` 或 `nav`），那么就使用这些替代 `section` 元素。
- ❖ `section` 元素有语义含义，意味着它所包含的内容具有相关性。如果您无法简洁地用几个词描述放在 `section` 元素中的内容，那么极有可能需要一个中立的容器替代它：更低级的 `div`。

也就是说，一般情况下，总是有语义的，所以在某些情况下，此解释是开放的。如果您可以提出充分的理由使用指定元素而不使用另一个，就替换它。万一有人要求您使用另一个，那么所引起的争论，对于所有参与的人来说，既具有娱乐性又内容丰富，甚至可能会得到关于规范的更广泛的社会共识。

请记住，如果使用得当，您也可以在现有的 `section` 元素中嵌套 `section` 元素。例如，对于在线新闻网站，世界新闻栏目可以按照全球各主要地区进行细分。

2.4.3 `article` 元素

`article` 元素与 `section` 元素类似，但也有一些明显的不同之处。以下是根据 WHATWG²给出的定义：

¹ <http://html5doctor.com/the-section-element/>

² <http://www.whatwg.org/specs/web-apps/current-work/multipage/sections.html#the-article-element>

`article` 元素表示在文档、页面、应用程序或网站中自我包含的部分，从原则上讲，是独立分布和重复使用的。例如，在企业联合组织中。

这一定义的关键术语是自我包含的部分和独立分配。`section` 元素可以包含任何主题的内容，`article` 元素必须是独立的一部分内容。这确实很难区分——所以在有疑问的时候，尝试一下这种联合测试：如果某一段内容可以无需修改重新发布在其他网站上，或通过 RSS 更新发布，或在社交媒体网站（比如 Twitter 或 Facebook）上推出，就说明它已经含有 `article` 元素。

最终，它是由您来决定如何构成 `article` 元素中的内容，这里有一些建议，仅供参考。

- ❏ 论坛帖子。
- ❏ 杂志或报纸文章。
- ❏ 博客条目。
- ❏ 用户提交的评论。

最终，就像 `section` 元素，`article` 元素可以被嵌套在其他 `article` 元素中。您也可以将 `section` 元素嵌套在 `article` 元素中，反之亦然。

2.4.4 nav 元素

可以安全地假设这个元素几乎会出现在每个项目中。`nav` 元素表示的就是它本身的含义：一组导航链接。尽管 `nav` 元素最普通的用途是包含一个无序链接表，但这里还有其他选择。例如，有一段文字包含有页面或页面某部分的主要导航链接，您也可以再为这一段文字添加 `nav` 元素。

在这两种情况下，`nav` 元素应为最重要的导航预留。所以，建议您避免在页脚的简短链接表中使用 `nav` 元素。



nav 元素及其辅助功能

您可能已经见过在许多网站上实行的设计模式“跳过导航”链接。这个想法可以使已经知道网站主要导航的屏幕阅读器用户跳过此处——毕竟，每一次都要听从大型网站的整个导航菜单，然后再单击它进入页面，这样做没有任何意义！

nav 元素有可能消除这种需要。如果一个屏幕阅读器看到 nav 元素，那么它可能允许用户跳过导航而无需任何其他链接。该规范规定：

用户代理（比如屏幕阅读器）的目标用户是：在最初呈现导航信息时已经被省略的用户；或者是可以立即使用导航信息的用户，他们可以使用该元素，作为一种决定页面哪些内容在最开始时被跳过或根据要求提供的一种方式。

当前屏幕阅读器无法识别导航时，这并不意味着您不应该使用它。辅助技术将继续得到发展，很可能您的网页将在以后的网络上呈现。现在通过建立标准，可以确保屏幕阅读器不断改进，您的网页会随着时间的推移变得更容易访问。



什么是用户代理？

在浏览规范时，您将会多次碰到术语“用户代理”。它仅是关于浏览器的一个非常具有想象力的术语——软件“代理”，就是用户用于进入页面内容的工具。规范不将它简单地称为“浏览器”是因为用户代理可以包括屏幕阅读器或其他阅读网页的技术方式。

您可以在指定页面使用多次 nav 元素。如果在您的网站上有一个主导航栏，那么在此处就需要一个 nav 元素。

另外，如果您有一组二级链接指向当前页面的不同部分（使用页锚点），那么这个工具也应包含在 nav 元素中。

与 section 元素一样，对它的使用仍有一些争论：由哪些构成 nav 元素的可接受用途，并且为什么在某些情况下不推荐使用（如在页脚内）。一些开发人员认为这种元素是适当的分页或主题链接路径下面的标题链接或搜索形式，这些构成了一个网站导航（就像在 Google 中）的主要手段。

这一决定最终将取决于您。WHATWG 的 HTML5 规范的首席编辑、程序开发人员 Ian Hickson 直接回答了这个问题：“使用它就像使用 class=nav 一样”。¹

2.4.5 aside 元素

此元素表示页面的某一部分与其他内容无关，将这部分内容放在 aside 元素

¹ 参见 <http://html5doctor.com/nav-element/#comment-213>

中，可认为它独立于其他内容¹。

`aside` 元素可用于包含与其他内容不相关的部分。

■ 特定独立的一部分内容（例如，文章或节）。

■ 作为惯用做法，将整个页面或文档作为“侧栏”，添加到网页或网站中。

`aside` 元素不应被用于包含页面中的主要内容部分。换句话说，它并不是附加说明。`aside` 元素可以自成一体，但仍是整体的一部分。

`aside` 元素的一些可行性用途包括侧边栏、二级链接表或广告区。应该注意的是，`aside` 元素（如在页眉的情况下）不是由它在页面的位置所决定的，它有可能在边上，也有可能在其他地方；它的内容本身及其与其他元素的关系，决定了它的位置。

2.4.6 footer 元素

我们将在本章讨论的最后一个元素是 `footer` 元素。与 `header` 元素一样，您可以在一个单独页面使用多个 `footer` 元素。您应使用 `footer` 元素包含页面的某一部分，也就是您通常在 `<div id="footer">` 中所包含的内容。

根据规范，`footer` 元素表示最近部分内容的页脚。“部分”内容可以是 `section`、`article` 或 `aside` 元素。

通常，页脚包含版权信息、相关链接列表、作者信息和您通常在一个内容块结尾处所添加的类似内容。然而，与 `aside` 和 `header` 元素一样，`footer` 元素没有根据其在页面上的位置对其进行定义。因此，它不必出现在某一部分的结尾处或页面的底部。当然最有可能会出现这些地方，但这不是硬性规定。例如，关于博客的作者信息可以在发布的文章的上方，而不是在下方，但这仍然被认为是页脚信息。



HTML5 创建者如何决定包含哪些新元素？

您可能想知道 HTML5 创建者如何想出了这些新语义元素。毕竟，您可以使用几十个切实的语义元素——但为什么没有用于用户提交评论的 `comment` 元素或专门用于广告的 `ad` 元素呢？

HTML5 的创作者通过运行测试，搜索数以百万的网页，看看最常使用哪些元素。根据检测元素的 `id` 和 `class` 属性来筛选这些元素。这一结果引导并帮助了

¹ <http://dev.w3.org/html5/spec/Overview.html#the-aside-element>

解新的 HTML 语义元素的创建。

因此，不会引进可能被拒绝或不被使用的新技术，HTML5 的编辑努力推出与网页作者和谐工作的元素。换句话说，对于常见的网页包含 id 为 header 的 div 元素很常见，这样包含一个 header 元素是非常有意义的。

2.5 构建 The HTML5 Herald

我们已经介绍了页面结构的基础以及在此方面非常有帮助的 HTML5 元素。现在我们将开始创建有内容的网页部分。

使用 header 元素，从顶部开始。首先在此包含徽标、名称以及标语。我们还可以在网站导航中添加 nav 元素。

在添加 header 元素后，网站的内容被分为 3 列。在您使用 section 元素时，可以停下来思考其内容。如果每一部分包含一个独立的“部分”信息（比如体育部分和娱乐部分），那将非常有意义。因为，分成独立的栏目仅是一个可视化的安排——所以在每一列，我们将使用普通的老元素 div。

在 div 里面，我们可以包含报纸文章。当然，这些也是 article 元素的候选者。

而在最右列，除了一篇文章，还有 3 个广告。我们将使用一个 aside 元素来包含广告，每一个广告放在 article 元素中。这似乎很奇怪，但回头看看文章描述：“一个自我包含的组成部分[...]也就是说，从原则上讲，是独立分布或可重复使用的。”一个广告与该海报完美匹配，因为它通常没有修改以在许多网站上转载。

下一步，我们将为出现在广告下面的最后一篇文章添加 article 元素。这最后一篇文章将不被包括在保存了 3 个广告的 aside 元素中。如果要使其归属于 aside 元素中，article 元素中的内容将需要与网页内容无关。但是事实不是这样的：article 元素内容是页面中主要内容的一部分，所以将它放在 aside 元素中是错误的。

现在，第三列由两个元素组成：aside 和 article 元素，其中一个放在另一个上面。为了将它们放在一起并容易设置其风格，我们将它们放在 div 元素中。我们在这里不使用 section 元素或其他语义标记，那样就意味着 article 和 aside 有某种局部的关系，但它们不是——在这里只是我们设计的一个功能，而它们恰巧

在同一列中而已。

在页眉下方，出于样式目的，还有一个完整的上半部分包含在通用的 `div` 元素中。

最后，我们在其传统的位置——页面底部添加 `footer` 元素。由于包含了一些不同的内容块，而每一个内容块都形成了自我包含和主题相关的单位，我们将这些分到 `section` 元素中。作者信息将形成一个部分，而每一个作者都在自己嵌套的 `section` 元素中。这样，另一个 `section` 元素放版权和其他信息。

让我们在页面添加新元素，文档如下：

index.html (excerpt)

```
<body>

<header>
  <nav></nav>
</header>

<div id="main">
  <div id="primary">
    <article></article>
    :
  </div>
  <div id="secondary">
    <article></article>
    :
  </div>
  <div id="tertiary">
    <aside>
      <article></article>
      :
    </aside>
    <article></article>
  </div>
</div><!-- #main -->

<footer>
  <section id="authors">
    <section></section>
  </section>

  <section id="copyright"></section>
</footer>

<script src="js/scripts.js"></script>
</body>
```

图 2.2 是一个屏幕截图，显示了一些页面标签，说明了我们所使用的一些主要结构元素。

我们现在已有了一个结构，可以作为内容的坚实基础。

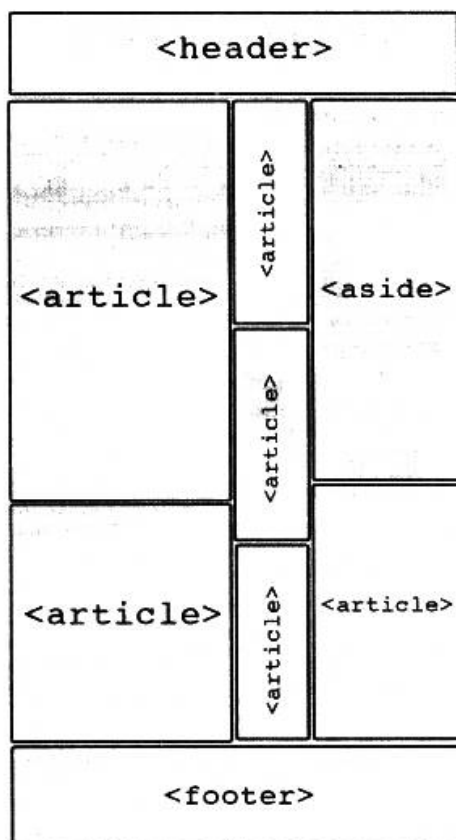


图 2.2 The HTML5 Herald 被分解为结构化的 HTML5 元素

2.6 小结

这就是本章的内容。我们已经学习了 HTML5 的内容结构基础，并且开始用我们所学到的知识创建了我们的示例项目。

在下一章中，我们将详细介绍在 HTML5 中如何处理不同类型的文章。然后，在处理更多新 HTML 元素时，将继续在我们的网页上添加语义元素。

关于 HTML5 语义的更多内容

我们的示例网站构建得很好。我们已经搭建了基本结构，并学习了使用 HTML5 的新元素标记内容的更多信息。

在本章中，我们将讨论更多的新元素，以及对一些熟悉的元素进行修改和改善。我们还将在项目中添加一些标题和基本文字内容，并讨论关于 HTML5 对于 SEO 和可访问性的潜在影响。

在我们进行深入讨论前，先介绍一下 HTML5 在表格方面带给我们的新并有点棘手的概念。

3.1 关于内容类型的新视角

出于布局和设置样式的目的，开发人员已经习惯于将 HTML 页面的元素归属于两类：块和内联。虽然元素仍被浏览器呈现为块和内联，但是 HTML5 规范会将内容进一步细分。目前该规范定义了更精细的**内容模型**。这些都是关于在指定元素中的内容的广泛定义。在大部分时间里，它们对您写的标记影响不大，但却应该熟悉它们，所以让我们快速浏览一下。

元数据内容

这一类听起来像是数据不在页面本身显示，但是影响页面的演示或包含其他页

面信息，包括 `title`、`link`、`meta` 及 `style` 等元素。

流内容

流内容包括几乎所有在 HTML 文档正文中使用的元素，包括 `header`、`footer` 以及 `p`，不包括那些不影响文档流程的元素（例如，在页面的头部的元素 `script`、`link` 和 `meta`）。

划分内容

这是 HTML5 中最有趣（最有用处、关系最密切）的内容类型。在最后一章，我们会发现自己经常使用通用术语“section”指代含有 `heading`、`footer` 或 `aside` 元素的内容块。事实上，我们实际指的是划分内容。在 HTML5 中，这一类包括 `article`、`aside`、`nav` 和 `section` 等元素。我们将谈论划分内容以及如何使您以详细、简洁的方式编写标记。

标题内容

这种内容类型定义了指定部分的标题，并包括各种层次的标题（`h1` 和 `h2` 等），以及新的 `hgroup` 元素。我们将在稍后详细介绍。

措辞内容

这一类大致相当于过去常使用的内联内容，包括 `em`、`strong`、`cite` 和 `like` 等元素。

嵌入内容

这一类相当易懂，包括那些嵌套在页面的元素，如 `img`、`object`、`embed`、`video` 和 `canvas` 等。

交互内容

这一类包括用户可以交互的任何内容。它主要由表单元素、链接以及在一定属性显示时进行交互的元素构成。

通过阅读上面的列表，正如您所看到的，某些元素可以属于多个类别，也有一些元素不属于任何类别。如果这看起来有些混乱，请不要担心：只要记住这些存在的差别，就已经绰绰有余。

3.2 文档大纲

在以前的 HTML 版本中，您可以通过查看包含在页面内不同等级的标题（h1～h6）来指定文档大纲。每次添加新级别的标题时，您就可以建立更深一层的结构大纲。例如，编写下列标记：

```
<h1>Title</h1>
:
<h2>Subtitle</h2>
:
<h3>Another level</h3>
:
<h2>Another subtitle</h2>
```

这将生成如图 3.1 所示的文档大纲。

最好是每页都有一个单独的 h1 元素，其他标题按顺序在下面排列。

为了使内容更容易联合（syndicate）和移动，HTML5 为构建 HTML 文档大纲提供了更清晰的算法。属于“划分内容”类别的每一个元素都在文档大纲中创建了新节点。在每一块划分内容的标题（h1～h6）元素也都创建了“暗含的”部分，这就是我们上面创建的简单大纲。

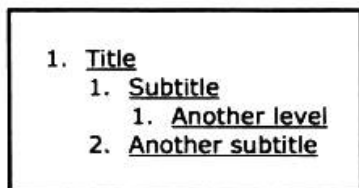


图 3.1 一个简单的文档大纲

这一切听起来好像更复杂了。为了开始能够理解它，让我们看一下如何使用另外一些 HTML5 元素重新编写上面的示例：

```
<section>
  <h1>Title</h1>
  :
  <article>
    <h1>Article Title</h1>
    :
    <h2>Article Subtitle</h2>
    :
  </article>
  <article>
    <h1>Another subtitle</h1>
    :
  </article>
</section>
```


得到的结果恰好与上面的文档大纲一样：每一部分划分内容（在本例中的 `article` 元素）都在文档树中创建了一个新节点，因此就有了自己的 `h1`。这样，每一部分都有了自己的微型文档大纲。

新大纲算法的优点是允许我们将整个部分移动到一个完全不同的文档中，并同时保持原有标记。在此之前，在专栏页面的专栏标题可能是 `h1`，但是在主页或类别分页的同样的专栏标题可能是 `h2` 或 `h3`。现在，只要一组标题集合在一个划分内容元素内，您就可以保留同样的标记。



测试文档大纲

在开始时，将 HTML5 文档大纲做好是一件棘手的事情。如果您有困难，可以使用一个非常方便的名为 `h5o1` 的 JavaScript 书签，它可以显示任何用 HTML5 大纲算法查看的文档大纲。其结果会根据 HTML5 标准显示文档的层次结构，因此您可以根据需要进行纠正。

要将它安装在您的浏览器上，从网站下载 HTML 文件，并在浏览器中打开它；然后将链接拖到您的收藏夹或书签栏中。现在您可以使用 `h5o` 链接来显示任何您正查看的网页的文档大纲。

请注意，有一点非常重要，编码和构建内容的旧方式（每一页都有一个单独的 `h1`）在 HTML5 中仍然有效。即使您错过了移动性和联合优势，您的网页继续有效。



理解划分根源

与划分内容相似，但是有区别。HTML5 也定义一种名为划分根源的元素。它包括 `blockquote`、`body`、`details`、`fieldset`、`figure` 和 `td`。是什么让划分根源元素与之不同？虽然它们可能有自己的大纲，但是在这些元素中的划分内容与标题都不属于整个文档的大纲（`body` 元素中的内容除外，它的大纲就是文档大纲）。

3.3 最新消息

现在我们已经掌握了 HTML5 中的内容类型和文档大纲的许多知识和技巧。现在让我们回到 The HTML5 Herald 并为文章添加一些标题。

¹ <http://code.google.com/p/h5o/>

为了简单起见，我们单独处理每一部分。为我们在导航上面的 header 部分添加标题和副标题。

```
<header>

  <hgroup>
    <h1>The HTML5 Herald</h1>

    <h2>Produced With That Good Ol' Timey HTML5 & CSS3</h2>
  </hgroup>
  <nav>
  :
  </nav>

</header>
```

hgroup 元素

您会注意到我们已经在标记中引入了 3 个元素：用习惯的 h1 元素标记的网站标题；用 h2 标记紧接在主页标题下方的宣传词；以及包含了标题和宣传词的新 HTML5 元素——hgroup。

为了理解 hgroup 元素的用途，请重新考虑一下如何创建页面的大纲。让我们看看没有 hgroup 元素的标题标记：

```
<h1>The HTML5 Herald</h1>
<h2>Produced With That Good Ol' Timey HTML5 & CSS3</h2>
```

这将生成文档大纲，如图 3.2 所示。

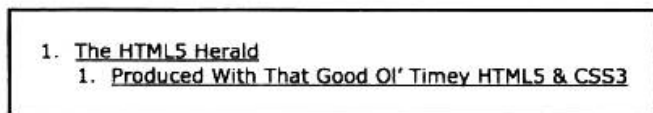


图 3.2 在文档大纲中，副标题生成了一个不需要的节点

h2 元素创建了一个新的、隐含的部分：所有内容都在由宣传词所创建的分区下的逻辑性分组中，这根本不是我们想要的。此外，如果我们想要另外一个使用 h2 的标题（例如，文章标题），这些新标题就会在层次上与宣传词同级；这也是不正确的，如图 3.3 所示。

那么，我们可以开始用 h3 标记随后的标题，对不对？但是，这再一次使我

们的文档大纲出现问题。现在，以 h3 开始的标题将成为我们宣传词的副主题，如图 3.4 所示。

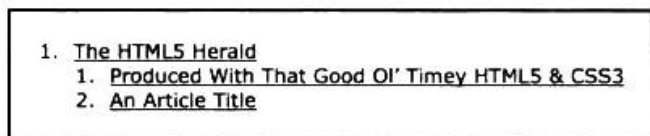


图 3.3 在内容中的其他标题错误地显示为与宣传词同组

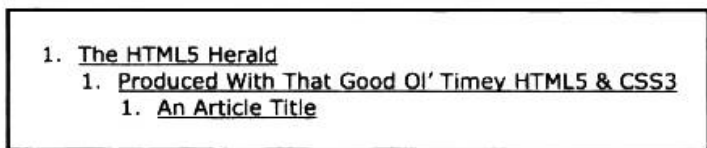


图 3.4 使用进一步嵌套标题等级未能解决问题

这也是不可取的，我们希望新的标题成为主要标题 h1 的附属部分。如果相反，我们选择用通用元素（比如 p 或 span）来标记我们的宣传词：

```
<h1>HTML5 Herald</h1>
<p id="tagline">Produced With That Good Ol' Timey HTML5 & CSS3
➡</p>
```

虽然这避免了用多余分支扰乱文档大纲，但还是在语义方面有所欠缺。您可能以为通过使用宣传词的值，id 属性会有助于定义元素的含义。但是 id 属性并不能通过浏览器来推断元素的含义，它对文档的语义没有丝毫的作用。

这就是要引入 hgroup 元素的地方。hgroup 元素告诉用户代理嵌套在里面的标题形成了一个复合标题（标题组），其中 h1 是主父元素。这就防止了让文档大纲变得混乱，并帮助我们在页面中避免使用非语义元素。

因此，在任何时候，如果您想添加副标题而不影响文档大纲，只要将标题放在 hgroup 元素中即可。这样既解决了问题，又无需借助于不良方法。图 3.5 显示了由标题生成的大纲，其中 hgroup 元素包含了两个标题。

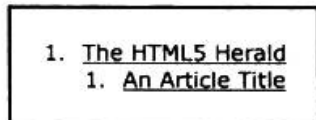


图 3.5 使用 hgroup 元素解决问题

这样就好了许多！

3.4 更多新元素

除了我们在第 2 章看到的结构元素以及刚才介绍的 `hgroup` 元素，HTML5 还引入了许多新语义元素。让我们来看看这些元素。

3.4.1 `figure` 和 `figcaption` 元素

HTML5 的新元素有助于更改语义，`figure` 和 `figcaption` 元素是另一对新的 HTML5 元素。在规范中对其解释如下：

该元素可以用于注释文章主要内容中的插图、图表、照片和代码清单等，但是它们在不影响文件流的情况下，可以移到其他主要内容中，例如，移到页面的一侧、专用页面或附录。

考虑到附在文章中的图表、图形和图像或示例代码。所有内容类型都是使用 `figure` 和 `figcaption` 元素的好地方。

为了使用 `figure` 元素，被放在 `figure` 元素中的内容必须与所放图形的主要内容相关联。如果可以将它从文档中全部删除，完全理解文档内容并不受影响，您可能不必使用 `figure` 元素：您可能需要使用 `aside` 元素或其他替代元素。同样，如果图像或清单形成了文档流，并且移到文本需要重新措辞，那么可能最好选用其他元素。

让我们看一下如何在 `article` 元素中标记 `figure` 元素。

```
<article>
  <hgroup>
    <h1>WAI-ARIA</h1>
    <h2>Web App Accessibility</h2>
  </hgroup>

  <p>Lorem ipsum dolor ... </p>

  <p>As you can see in <a href="#fig1">Figure 1</a>,

  <figure id="fig1">
    <figcaption>Screen Reader Support for WAI-ARIA</figcaption>
    
  </figure>
```

```
<p>Lorem ipsum dolor ... </p>
</article>
```

3.4.2 mark 元素

mark 元素“表示文档的一部分由于可能与用户当前活动相关，已被突出显示。”不可否认，极少情况下我们能想到使用 mark 元素。最常见的是在搜索文章时，搜索关键词在搜索结果中被突出显示。

避免 mark 元素与 em 或 strong 元素混淆；这些元素都添加了语境的重要性，而 mark 元素是以用户当前浏览或搜索行动为根据，对目标内容进行分离。

例如，一个用户在谷歌中搜索关键词“HTML5”后，转到您网站中的某一篇文章，您可能使用 mark 元素在文章中突出显示所有的“HTML5”一词。代码如下：

```
<h1>Yes, You Can Use <mark>HTML5</mark> Today!</h1>
```

可以使用两种方式将 mark 元素添加到文档中：使用服务器端代码；在页面加载时，使用 JavaScript。

3.4.3 progress 和 meter 元素

在 HTML5 中新增添的两个元素可以用来标记以某种方式测量和衡量的数据。它们之间的区别相当微妙：progress 元素用来描述程序进展过程完成的当前状态，无论完成状态是否被定义。传统的下载进度条就是 progress 元素的完美示例。

meter 元素表示一个已知其范围的元素，这意味着它具有明确的最大、最小值。规范给出了磁盘使用或投票人数比例的示例，两者都有明确的最大值。因此，很可能您不会使用 meter 元素表示年龄、身高或体重，因为通常这些值没有最大值。

让我们首先来看看 progress 元素。progress 元素可以具有 max 属性以表示任务完成的终点，并且使用 value 属性来表示任务状态。这两个属性都是可选的。以下是一个示例：

```
<h1>Your Task is in Progress</h1>
<p>Status: <progress min="0" max="100" value="0"><span>0</span>%
</progress></p>
```


该元素最好用于（与一些 JavaScript 连同使用）动态改变百分比值，例如，任务进展。您会注意到，代码包含的标签隔离了数值；在您需要更新此数值时，它有助于在脚本中直接找到该目标数值。

meter 元素有 6 个相关属性。除了 max 和 value 属性外，还可以使用它的 min、high、low 和 optimum 属性。

min 和 max 属性为范围的上下限提供了参考数值，value 属性显示的是当前的额定数值。high 和 low 属性表明的是在文章中被分为的“高”和“低”的阈值。例如，考试成绩的范围可以从 0%到 100%（最大），但是任何低于 60%的值都被认为是低值，任何高于 85%的值都被认为是高值。optimum 指的是理想值。在考试得分的情况中，optimum 的值是 100。

下面是一个使用 meter 元素的示例，假设以磁盘使用为例：

```
<p>Total current disk usage: <meter value="63" min="0" max="320"
  ↳low="10" high="300" title="gigabytes">63 GB</meter>
```

3.4.4 time 元素

日期和时间是非常宝贵的网页组件。搜索引擎能够根据时间筛选结果，在某些情况下，搜索算法可以根据第一次发布的时间，决定特定的搜索结果所接收到的或多或少的权重值。

time 元素被专门设计用来处理人与机器读取时间和日期有误差的问题。看看下面的示例：

```
<p>We'll be getting together for our next developer conference on
  ↳12 October of this year.</p>
```

当读到本段时，人们将理解什么时候会发生这个事件，但是机器要解析这个信息，显然不够清楚。

下面是引入了 time 元素的相同段落：

```
<p>We'll be getting together for our next developer conference on
  ↳<time datetime="2011-10-12">12 October of this year</time>.</p>
```

在 datetime 属性中保留明确的日期和时间，time 元素就可以以任何您喜欢的格式显示日期和时间。尽管当前没有浏览器可以专门处理 time 元素，此值仍可

以用 JavaScript，或通过浏览器本身，将此值转换为局部的或优先的。

如果您想在包括时间的时候也包括日期，请执行下面的代码：

```
<time datetime="2011-10-12T16:24:34.014Z">12 October of this year.
➡</time>
```

在上面的示例中，T 字符用来表示时间的开始。格式为 HH:MM:SS 以及小数点后的毫秒。Z 字符是可选的，并且表示该时区是世界标准时间（UTC）。为了表示时区偏移量（代替 UTC），您需要追加一个加号或减号，就像这样：

```
<time datetime="2011-10-12T16:24:34.014-04:00">12 October of
this year</time>
```

除 datetime 属性显示在上面的示例中，time 元素也允许使用 pubdate 属性。这是一个布尔型属性，它的存在表明了嵌套在最近 article 元素的内容是在指定日期发布的。如果没有 article 元素，pubdate 属性将作用于整个文档。

例如，在 The HTML5 Herald 的页眉部分，本期的出版日期是 time 元素和 pubdate 属性的最佳候选内容：

index.html (excerpt)

```
<p id="issue"><time datetime="1904-06-04" pubdate>June 4, 1904
➡</time></p>
```

此元素表明了报纸的出版日期，我们添加了 pubdate 属性。此页面的其他日期（例如，在文章中）将忽略此属性。

time 元素有一些相关规定和准则：

- ❶ 不应使用 time 元素编码未指定的日期和时间（例如，“在冰河时代”或“去年冬天”）。
- ❷ 所表示的时间不能是“BC”或“BCE”（公元前），必须是公历日期。
- ❸ datetime 属性必须是有效的日期字符串。
- ❹ 如果 time 元素缺乏 datetime 属性，那么元素的文本内容（无论怎样，要出现在开始和结束标记之间）需要是一个有效的日期字符串。

time 元素的用途是数之不尽的：日历事件、出版日期（博客、视频、新闻发

布等)、历史日期、交易记录、文章或内容更新,还有更多。

3.5 对现有功能的更改

虽然新元素和 API 是 HTML5 最主要的内容,但是最新一代的网络标记也给现有的元素带来了改变。在大多数情况下,任何更改都已经考虑了它们的向后兼容性,以确保现有内容的标记仍可使用。

我们已经介绍了一些更改(例如,文档类型声明、字符编码、内容类型和文档大纲)。让我们看看 HTML5 规范所引入的一些其他重大更改。

3.5.1 单词“Deprecated”是被弃用的

在 HTML 和 XHTML 的早期版本中,不再被推荐使用的元素(已从规范中删除的)被认为是“弃用的”。在 HTML5 中,不再有任何被弃用的元素,现在使用的术语是“过时”。

这看起来是微不足道的变化,但是这个区别是非常重要的:弃用的元素从规范中删除了,过时的元素还保留在那里。所以,即使已不再推荐使用这些元素,浏览器制造商仍然采用这个标准方式始终如一地呈现它们。例如,您可以登录 <http://dev.w3.org/html5/spec/Overview.html#frames>,在 W3C 规范中查阅关于帧的信息(一个过时的功能)。

3.5.2 链接中的 block 元素

虽然过去大多数浏览器已经能够较好地处理此问题了,但是从来没有真正有效地将一个块级的元素放在一个元素内。相反,为了生成一个有效的 HTML,您必须使用多个 a 元素,并统一组的风格,使其显示为一个单一的块。

在 HTML5 中,您现在可以包含几乎所有内容——而不需要表单元素和其他链接——只要在 a 元素中,就无需担心验证错误。

3.5.3 黑体文本

已经更改了在 HTML5 中语义定义黑体文本的方式。在大多数浏览器中使文本为黑体的最基本的两种方式是:使用 b 元素,或使用 strong 元素。

尽管 `b` 元素从未被弃用，但是在 HTML5 之前，就更赞成使用 `strong` 元素了。`b` 元素以前被称为“使这段文字呈现粗体”的一种方式。由于 HTML 标记是关于内容的意义，留给 CSS 处理呈现方式，这显然是不能令人满意的。

在 HTML5 中，重新定义了 `b` 元素，使之表示为一段文字，即“从正常的文章文体偏移而无需传递任何其他重要信息。”

`strong` 元素此时还或多或少传达了相同的含义。在 HTML5 中，它表示“内容非常重要。”有趣的是，HTML5 规范允许嵌套 `strong` 元素。因此，如果整个句子由一条重要的警告组成，但是某些词更加重要，那么这句话可以被包含在 `strong` 元素中，每一个重要的词可以嵌套在自己的 `strong` 元素中。

3.5.4 斜体文本

随着对 `b` 元素和 `strong` 元素进行修改，在 HTML5 中更改了 `i` 元素的定义方式。

以前，`i` 元素被简单地用于显示斜体文本。就像 `b` 元素一样，这个定义并不能令人满意。在 HTML5 的定义中已被更新为“在另一种语态或基调，或从正常文体偏移的另一种方式的一段文本。”因此，虽然文本仍然显示为斜体，但是文本的呈现与语义没有任何关系——这取决于您。

现在来看一个示例，用 `i` 标签偏移的那部分内容可能是另一种语言的惯用词语，如“*reductio ad absurdum*”，这个拉丁短语的意思是“缩减荒谬的程度。”还有一些其他的示例：在小说作品中代表一段梦境的文字，或者是杂志文章的物种学名。

`em` 元素没有被更改，但它的定义已经扩大，使其用处变得更加清楚。它仍然指所强调的文字，就和我们口语对话中的情形一样。例如，下面两个短句具有完全相同的措辞，在不同的地方使用 `em` 元素，它们的含义也不同：

```
<p>Harry's Grill is the best <em>burger</em> joint in town.</p>
<p>Harry's Grill <em>is</em> the best burger joint in town.</p>
```

在第一个句子里，由于强调的是单词“*burger*”，因此句子的意思强调的是正在讨论的“*joint*”类型。在第二句中，强调的是单词“*is*”，这样，句子的侧重点就转移到了哈利的烤肉是否真的是城里最好的汉堡店。

无论是 `i` 元素还是 `em` 元素，都不应用于标记出版物标题；而是应该使用 `cite` 元素（请参见“引起争议的 `cite` 元素”一节）。

这里讨论的所有 4 个元素（`b`、`i`、`em` 和 `strong`）中，唯一对其内容给出语义

重要性的是 `strong` 元素。

3.5.5 大号和小号文体

`big` 元素以前用来表示大字体显示的文本。`big` 元素现在已经过时并不再被使用。但是 `small` 元素仍有效，且具备不同的含义。

此前，`small` 元素的目的是描述“小字体的文本。”在 HTML5 中，它表示“页边注释，例如小号字体印刷。”在一些示例中，`small` 元素可用于包含页脚文字、细字印刷、条款协议等信息。`small` 元素应仅用于短版印刷的文字。

虽然从定义里删除了 `small` 元素的呈现含义，但是在 `small` 标签中的文字仍有可能显示为比文档的其他文字要小的字体。

例如，The HTML5 Herald 页脚部分包含的版权声明。实质上这是法律细字印刷，用 `small` 非常完美：

```
<small>&copy; SitePoint Pty. Ltd.</small>
```

3.5.6 引起争议的 `cite` 元素

`cite` 元素是另一个在 HTML5 中被重新定义的元素，并且引起了一些争议。在 HTML4 中，`cite` 元素表示“引用或参照其他资源。”在该定义范围内，规范允许将人的姓名标记在 `cite` 元素中（例如，在引用个人信息的情况下）。

HTML5 明文禁止使用 `cite` 引用人的姓名，这似乎违背了向后兼容的原则。现在，规范将 `cite` 元素描述为“一个作品的标题”，并给出许多示例，包括书、歌曲、电视节目和戏剧作品。

一些著名的 Web 标准倡导者（包括 Jeremy Keith 和 Bruce Lawson）曾反对新定义禁止在 `cite` 元素中引用人的姓名。关于更多正在进行的辩论资料，请参阅 WHATWG Wiki¹ 网站的专题页面。

3.5.7 描述（不是定义）列表

在 HTML5 规范中，现有的 `dl`（定义清单）元素以及与之相关的 `dt`（术语）元素和 `dd`（描述）子级都已被重新定义。以前，除了术语和定义外，规范还允许

¹ http://wiki.whatwg.org/wiki/Cite_element

dl 元素标记对话，但是现在规范禁止了此用法。

在 HTML5 中，这些列表已不再称为“定义列表”，现在使用更通用的叫法，即“描述列表”。它们应被用于标记任何名值对，包括术语和定义、元数据标题和值，以及问题和答案。

3.6 其他新元素及功能

我们已经向您介绍并详细说明了一些非常实用的新元素以及功能。现在，在这一节中，我们将介绍那些新添加在 HTML5 规范中鲜为人知的元素、属性及功能。

3.6.1 details 元素

这种新元素有助于标记文档的隐藏部分，而且可以扩展显示附加信息。该元素的目的是为网络上的共同功能提供全面支持——一个有标题的可分解的格子，以及更多的隐藏信息和功能。

通常使用结合标记和脚本创建这种类型的构件。在 HTML5 中，将删除脚本要求，并为 Web 设计人员简化执行程序。

代码如下：

```
<details>
  <summary>Some Magazines of Note</summary>
  <ul>
    <li><cite>Bird Watchers Digest</cite></li>
    <li><cite>Rowers Weekly</cite></li>
    <li><cite>Fishing Monthly</cite></li>
  </ul>
</details>
```

上面的示例会将 summary 元素中的内容以及其他隐藏的内容呈现给用户。只要单击 summary 元素，即可显示隐藏内容。

如果 details 元素中没有定义的 summary 元素，用户代理将定义一个默认的 summary 元素（例如，“Details”）。如果您想将隐藏内容默认为可视，那么您可以使用布尔型 open 属性。

summary 元素仅可被用于 details 元素的子代，并且必须是第一子代。

到目前为止, details 元素几乎得到了所有浏览器的支持。许多基于 JavaScript 的补充支持可供选择, 其中包括由 Mathias Bynens¹所提供的。

3.6.2 自定义的有序列表

使用 ol 元素的有序列表在 Web 网页中十分常见。HTML5 引入了一个新的布尔型属性, 称为 reversed。当使用此属性时, 将会反转列表项目的顺序。

当我们处理有序列表题目时, HTML5 恢复了在 HTML4 中弃用的 start 属性。start 属性指定列表起始的序号。

对 start 属性的支持很好, 但是大多数浏览器都不支持 reserved 属性。

3.6.3 作用域样式

style 元素, 在网页上直接用于嵌入文本样式, 现在允许使用一个称为 scoped 的布尔型属性。示例代码如下:

```
<h1>Page Title</h1>
<article>
  <style scoped>
    h1 { color: blue; }
  </style>
  <h1>Article Title</h1>
  <p>Article content.</p>
</article>
```

由于代码中存在 scoped 属性, 因此在 style 元素中的文本样式仅应用于父代元素及其子代元素 (如果允许级联原则), 而不是整个文档。这就允许文档的特定部分 (如上面示例中的 article 元素中的内容) 可以很容易地与它们相关联的样式一起移植。

这当然是一个方便的新功能, 但是截止到编写本书时, 还没有浏览器支持 scoped 属性。作为一个临时解决方案, 在 [https://github.com/thingsinjars/jquery-Scoped-CSS-plugin](https://github.com/thingsinjars/jquery-scoped-css-plugin) 中可提供一个基于 jQuery 的补充支持。

3.6.4 script 元素的 async 属性

script 元素允许使用 async 属性, 此属性类似于现有的 defer 属性。使用

¹ <http://mathiasbynens.be/notes/html5-details-jquery>

`defer` 属性规定了浏览器需等候，直到加载脚本之前、页面标记被解析。而新的 `async` 属性允许您指定脚本异步加载（意味着只要可以提供脚本，便可加载）而不会造成页面其他元素在加载时延迟。`defer` 和 `async` 都是布尔型属性。

这些属性仅用于 `script` 元素定义外部文件。对于旧的浏览器，如果有必要，您可以包含 `async` 属性和 `defer` 属性，这样可以确保使用其中的一个。在实践中，两个属性在加载脚本时，都不会暂停浏览器呈现页面。然而，`async` 属性的好处往往更多，当其他呈现任务执行时，它“在后台”加载脚本，并在脚本可用时马上执行脚本。

如果您下载的脚本没有其他依赖关系，那么异步属性特别有用。此外，如果脚本能够迅速加载，而不是在页面加载后，那么将有益于用户体验。

3.7 验证 HTML5 文档

在第 2 章中，我们向您介绍了 HTML5 中的许多语法更改，并且接触到了一些关于验证的问题。下面让我们将这些概念再展开一些，以便更好地理解验证页面是如何被更改的。

HTML5 的验证程序不再限制代码风格。您可以使用大写字母、小写字母、省略属性的引号，使标签打开，并且前后不一致，即使这样，您的页面通常仍然有效。

所以，您会问，什么才算是 HTML5 验证错误呢？验证程序会提醒您元素使用不当，其中包括使用的地方不恰当，丢失所需属性，不正确的属性值等类似错误。简单地讲，验证程序将让您知道使用的标记与规范是否冲突，所以，当开发网页时，它是一个极其有价值的工具。

然而，由于我们已经习惯于更严格的应用于 XHTML 文档的验证规则，因此，让我们讨论一些具体细节。这样，您就可以理解在 HTML5 中什么是被认为有效的，但在检查基于 XHTML 的页面时是无效的。

- 文档在进行 HTML5 验证时，不再需要那些基于 XHTML 的语法中所需要的元素；示例包括 `html` 和 `body` 元素。
- 虚元素或独立并不包含任何内容的元素不再要求使用关闭斜线；示例包括 `<meta>` 和 `
`。
- 元素和属性可以是大写、小写，或大小写混用。
- 除非是多个空格分隔的值、或者是 URL 显示为一个值并且包含带有等号

(=) 的查询字符串，属性值不必使用引号。

- 一些在基于 XHTML 语法所要求的属性已不在 HTML5 中要求；示例包括 script 元素中的 type 属性，以及 html 元素的 xmlns 属性。
- 一些被弃用并且在 XHTML 中无效的元素，现在又称为有效的：例如，embed 元素。
- 不在任何元素中的分散文字将会使 XHTML 文档无效；在 HTML5 中不会有此类情况。
- 在 XHTML 中需要关闭的元素，可以在 HTML5 中保持打开而不会引起验证错误；例如，p、li 和 dt。
- form 元素不再要求有 action 属性。
- form 元素中，例如，input 元素可以被当作 form 元素的直接子元素；在 XHTML 中，要求另一个元素（例如 fieldset 元素或 div 元素）包含 form 元素。
- textarea 元素不再要求有 rows 和 cols 属性。
- 在 XHTML 中弃用并无效的 target 属性，在 HTML5 中有效。
- 在 a 元素中可放置 block 元素。
- 表示“和”的符号（&）如果出现在页面的文章中，无需将它编码为&。

虽然难以面面俱到，但这是一个相当全面的、说明 XHTML 和 HTML5 验证区别的列表。有一些是格式选择，您应选择一种样式并保持一致。我们已经在前面的章节中介绍了一些比较好的样式选择，如果在您的 HTML5 项目中没有考虑到这些因素，欢迎采纳这些建议。



lint 工具

如果想使用更严格的准则验证标记语法样式，则可以使用 HTML5 lint 工具，例如，<http://lint.brihten.com/html/>。在编写本书时，它仍处于开发阶段，但已运行良好。您可以使用它检测属性及标签采用的是否是小写字母，空标签是否自我关闭的，布尔型属性忽略了其数值，从未省略结束标记——或这些样式原则的组合使用。它甚至可以确保您的标记一致缩进。

3.8 小结

现在，我们已经了解了 HTML5 中对新语义和语法进行更改的所有内容。其中一些信息可能有点难于立即理解，但是不要担心！熟悉 HTML5 最好的办法就是使用它，您可以从下一个项目开始就使用它。尝试使用一些我们在上一章介绍的结构元素，或者是在本章中介绍的一些文本级语义。如果您不确定究竟如何使用某个元素，再回去阅读相关部分，或者可以直接阅读规范。虽然语言肯定比本书枯燥（至少，我们希望它是这样！），但是规范可以使您更全面地了解指定元素如何使用。请记住，HTML5 规范仍在开发中，因此我们所介绍的内容仍可能发生改变。规范将始终包含最新信息。

在下一章中，我们将介绍 HTML5 引入的新功能的关键部分：表单及表单相关的功能。

HTML5 表单

我们已经为页面的绝大部分编写了程序，并且您现在已经知道很多关于新 HTML5 元素及相关语义的内容。但是，在我们开始看这个网站前（我们在第 6 章详细介绍），先快速跳过 The HTML5 Herald 的头版，看一下注册页面。这里会说明 HTML5 在 Web 表单方面所提供的內容。

HTML5 Web 表单引入了新的表单元素、输入类型、属性和其他功能。有些功能已经使用了多年：表单验证、组合框和占位符文本等。不同之处在于，之前我们必须求助于 JavaScript 创建这些功能，现在可以在浏览器中直接使用它们；您需要的就是在标记中将属性设置为可用。

HTML5 不仅使开发人员更容易制作表单，同时也为用户带来了许多方便。由浏览器处理客户端的验证，这样使不同的网站具有了更大的一致性，许多网页无需加载多余的 JavaScript，从而使速度更快。

下面让我们进入正题！

4.1 工具箱中的相关工具

将表单包含在页面中通常是开发人员最后做的事情——一些开发人员发现表单平淡乏味。好消息是，HTML5 在对表单进行代码的过程中注入了一些乐趣。在本

章最后，我们期待您能够在标记中恰当地使用表单元素。

让我们从普通的、老式的 HTML 注册表单开始：

register.html (excerpt)

```
<form id="register" method="post">
  <hgroup>
    <h1>Sign Me Up!</h1>
    <h2>I would like to receive your fine publication.</h2>
  </hgroup>

  <ul>
    <li>
      <label for="register-name">My name is:</label>
      <input type="text" id="register-name" name="name">
    </li>
    <li>
      <label for="address">My email address is:</label>
      <input type="text" id="address" name="address">
    </li>
    <li>
      <label for="url">My website is located at:</label>
      <input type="text" id="url" name="url">
    </li>
    <li>
      <label for="password">I would like my password to be:</label>
      <p>(at least 6 characters, no spaces)</p>
      <input type="password" id="password" name="password">
    </li>
    <li>
      <label for="rating">On a scale of 1 to 10, my knowledge of
      ➤HTML5 is:</label>
      <input type="text" name="rating" id="rating">
    </li>
    <li>
      <label for="startdate">Please start my subscription on:
      ➤</label>
      <input type="text" id="startdate" name="startdate">
    </li>
    <li>
      <label for="quantity">I would like to receive <input
      ➤type="text" name="quantity" id="quantity"> copies of <cite>
      ➤The HTML5 Herald</cite>.</label>
    </li>
    <li>
      <label for="upsell">Also sign me up for <cite>The CSS3
      ➤Chronicle</cite></label>
```

```

        <input type="checkbox" id="upsell" name="upsell">
    </li>
    <li>
        <input type="submit" id="register-submit" value="Send Post
        ↪Haste">
    </li>
</ul>
</form>

```

这个示例中的注册表单，在 HTML 最早版本中已可用。这种表单通过 label 和 p 元素向用户提示每个信息栏的输入数据类型，因此即使使用 Netscape 4.7 和 Internet Explorer 5（开玩笑！）的用户也能够理解此表单。它可以运行，但应得到改进。

在本章中，我们将使用 HTML5 的功能，从而使表单得到加强。HTML5 特别针对 email 地址、URL、数字和日期等提供了新的输入类型。除了这些新的输入类型，HTML5 也引入了可以与新的和目前使用的输入类型一起使用的属性。这些允许您提供所需的占位符文本、标记栏，以及声明可接受的数据类型——这些都未用 JavaScript。

稍后在本章中我们将向您介绍所有新添加的输入类型。在开始之前，让我们先看一下 HTML5 提供的新表单属性。

4.2 HTML5 表单属性

多年以来，开发人员已经编写了一些 JavaScript 代码段，用来验证用户输入表单栏的信息：需要什么元素和接受的数据类型等。HTML5 为我们提供了一些属性，从而允许我们规定什么是可接受的值，并通知用户输入了错误信息等，这些都无需使用 JavaScript。

支持这些 HTML5 属性的浏览器会将用户输入的数据与开发人员（您）提供的常规表达模式进行比较。然后检查是否确实填写了所有所需栏目，如果允许，可使用多个值等。更好的是，包含这些属性将不会影响到旧版浏览器：旧版浏览器将会忽略所有不兼容的属性。实际上，您可以使用这些属性和值改进脚本运行效率，而不需要将验证模式硬编码到您的 JavaScript 代码中，或在标记中添加多余的类。我们稍后将介绍如何完成这些内容，现在，让我们先详细讨论一下每个新属性。

4.2.1 required 属性

布尔型 required 属性告诉浏览器只有正确填写了问题字段，才提交表单。显

然，这意味着问题字段不能是空的，但也意味着根据其他属性或字段类型，只接受某些类型的值。在本章后面的章节中，我们将介绍几种不同的方式，使浏览器知道在表单中可以接受哪种类型的数据。

如果所需字段为空或无效，表单将无法提交，而且光标将移到第一个无效表单元素。Opera、FireFox 和 Chrome 浏览器将为用户提供错误信息，例如，如果是空值，浏览器就会提示“请填写此字段”或“必须指定值”，或者在数据类型或模式错误时，提示“xyz 不符合此页的格式要求”。



失去焦点？

让我们快速复习一下：当用户使用鼠标单击字段或在键盘上敲击 tab 键时，焦点就对准了表单元素。对于 input 元素，使用键盘打字便会将数据输入到那个元素中。

在 JavaScript 术语中，当 focus 事件接收到焦点时，它会触发表单元素；当失去焦点时，就会触发 blur 事件。

在 CSS 中，:focus 伪类可用于设置目前处于焦点的元素的样式。

除了在通常已有默认值的 button、range、color 和 hidden 元素中，required 属性可以设置任何输入类型。正如我们日前所见到的其他布尔型属性一样，如果您使用 XHTML，其语法可以是简单的 required 或 required="required"。

让我们在注册表单中添加 required 属性。我们将姓名、email 地址、密码以及订购起始日期字段设置为必需的。

register.html (excerpt)

```
<ul>
  <li>
    <label for="register-name">My name is:</label>
    <input type="text" id="register-name" name="name"
    ➤required aria-required="true">
  </li>
  <li>
    <label for="email">My email address is:</label>
    <input type="text" id="email" name="email"
    ➤required aria-required="true">
  </li>
  <li>
    <label for="url">My website is located at:</label>
```

```

    <input type="text" id="url" name="url">
</li>
<li>
    <label for="password">I would like my password to be:</label>
    <p>(at least 6 characters, no spaces)</p>
    <input type="password" id="password" name="password"
➤required aria-required="true">
</li>
<li>
    <label for="rating">On a scale of 1 to 10, my knowledge of
➤HTML5 is:</label>
    <input type="text" name="rating" type="range">
</li>
<li>
    <label for="startdate">Please start my subscription on:
➤</label>
    <input type="text" id="startdate" name="startdate"
➤required aria-required="true">
</li>
<li>
    <label for="quantity">I would like to receive <input
➤type="text" name="quantity" id="quantity"> copies of <cite>
➤The HTML5 Herald</cite></label>
</li>
<li>
    <label for="upsell">Also sign me up for <cite>The CSS3
➤Chronicle</cite></label>
    <input type="checkbox" id="upsell" name="upsell">
</li>
<li>
    <input type="submit" id="register-submit" value="Send Post
➤Haste">
</li>
</ul>

```

为了提高可访问性，在需要包含 `required` 属性时，请添加 **ARIA** 属性 `aria-required="true"`。一些屏幕阅读器缺乏对 **HTML5** 新功能的支持，但是很多都支持 **WAI-ARIA** 任务，所以添加此任务可以使用户知道该字段是必须填写的，在附录 B 中简要介绍了 **WAI-ARIA**。

图 4.1、图 4.2 和图 4.3 显示了在提交表单时，`required` 属性的运行状态。

设置必填表单字段的样式

您可以使用 `:required` 伪类设置必填表单元素的样式。您也可以使用 `:valid` 和 `:invalid` 伪类设置有效和无效字段的样式。用这些伪类和一些 **CSS** 魔法，您可

以为用户提供视觉线索，指出哪些字段是必须填写的，并且也对数据成功输入进行反馈。

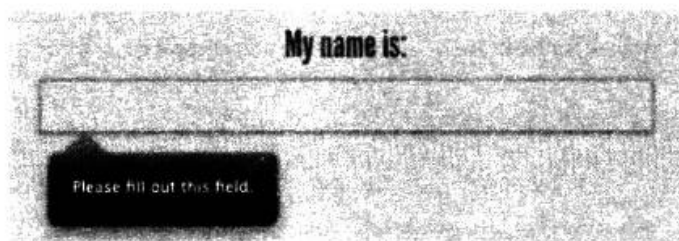


图 4.1 在 Firefox 4 中所需填写字段的验证信息

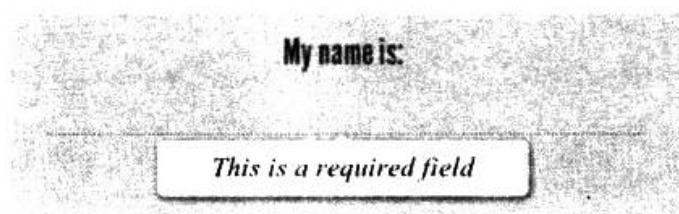


图 4.2 在 Opera 中的显示效果

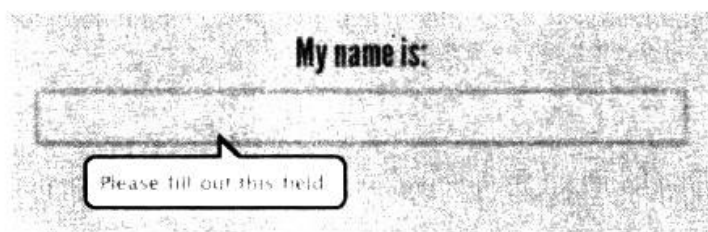


图 4.3 在 Google Chrome 中的显示效果

```
input:required {
    background-image: url('../images/required.png');
}
input:focus:invalid {
    background-image: url('../images/invalid.png');
}
input:focus:valid {
    background-image: url('../images/valid.png');
}
```

我们在所需填写表单字段添加背景图像（星号）。我们还为有效和无效字段添加单独的背景图像。只有在焦点对准表单元素时，此变化才非常明显，这样能防止表单看上去杂乱无章。



小心默认样式

请注意，Firefox 4 对于无效元素应用自己的样式（红色阴影），如上面图 4.1 所示。您也许想删除这种自带的投影，可以使用下面的 CSS 代码：

```
:invalid { box-shadow: none; }
```



向后兼容性

旧版浏览器也许不支持 `required` 伪类，但是您仍可使用属性选择符提供目标样式：

```
input:required,
input[required] {
    background-image: url('../images/required.png');
}
```

您也可以在不支持 HTML5 的浏览器中使用这个属性作为表单验证的一种方法。JavaScript 代码可以检查空元素的 `required` 属性是否存在，如果发现，将不能提交表单。

4.2.2 placeholder 属性

在表单元素中，`placeholder` 属性允许显示简短的提示，如果空间允许，将告诉用户在字段中应输入什么数据。在字段获得焦点时，占位符文本消失，如果处在模糊状态没有数据输入时，那会占位符文本会再次出现。开发人员多年来一直采用 JavaScript 来提供此功能，在 HTML5 中，占位符属性是自带的，而不再需要 JavaScript。

在 The HTML5 Herald 注册表中，我们将占位符添加到网站 URL 和开始日期字段中：

register.html (excerpt)

```
<li>
    <label for="url">My website is located at:</label>
    <input type="text" id="url" name="url"
    ➤placeholder="http://example.com">
</li>
```

```

:
<li>
  <label for="startdate">Please start my subscription on:</label>
  <input type="text" id="startdate" name="startdate" required
  ➔aria-required="true" placeholder="1911-03-17">
</li>

```

由于只有最新版的浏览器才能够支持 `placeholder` 属性，因此您不能将它当作通知用户需求的唯一方式。如果您的提示超过了字段的长度，那么可以在输入的 `title` 属性或下一个 `input` 元素的文字中描述该要求。

目前，Safari、Chrome、Opera 和 Firefox 4 都支持 `placeholder` 属性。

用 JavaScript 补充支持

与本章的其他内容一样，它不会影响到那些不支持 `placeholder` 属性的浏览器。

与 `required` 属性一样，您可以使用 JavaScript 魔法使旧版浏览器获得支持，从而可以使用 `placeholder` 属性和它的值。

处理方法：首先，使用 JavaScript 判定哪些浏览器缺乏支持。然后，在那些浏览器中，使用函数创建伪占位符。函数需要判定哪个表单字段包含 `placeholder` 属性，然后临时抓取属性内容并将它放在 `value` 属性中。

然后，您需要创建两个事件处理程序：一个是清除焦点所在的字段值；另一个是如果表单控件的值是 `null` 或一个空字符串，取代模糊状态的占位符值。如果您使用这个方法，就要确保 `placeholder` 属性值不是用户实际要输入的值，并在表单提交时清除伪占位符。否则，您将有許多诸如“XXX-XXXX”此类的提交！

让我们来看一个 JavaScript 代码片段示例（为简单起见，我们使用 jQuery JavaScript 库），从而使用 `placeholder` 属性逐步改进我们的表单元素。



jQuery

在下面的代码示例中，我们将使用 jQuery¹ JavaScript 库，并贯穿于本书的其余部分。虽然我们所添加的效果都能够使用普通的 JavaScript 来完成，但是我们发现 jQuery 代码更具有可读性；因此，它将有助于演示重点要介绍的内容（HTML5 API），而不是用很多时间解释 JavaScript。

¹ <http://jquery.com/>

下面是关于 placeholder 的代码:

register.html (excerpt)

```
<script>
if(!Modernizr.input.placeholder) {

    $("input[placeholder], textarea[placeholder]").each(function() {
        if($(this).val()==""){
            $(this).val($(this).attr("placeholder"));
            $(this).focus(function(){
                if($(this).val()==$(this).attr("placeholder")) {
                    $(this).val("");
                    $(this).removeClass('placeholder');
                }
            });
            $(this).blur(function(){
                if($(this).val()==""){
                    $(this).val($(this).attr("placeholder"));
                    $(this).addClass('placeholder');
                }
            });
        }
    });

    $('form').submit(function(){
        // first do all the checking for required
        // element and form validation.
        // Only remove placeholders before final submission
        var placeheld = $(this).find('[placeholder]');
        for(var i=0; i<placeheld.length; i++){
            if($(placeheld[i]).val() ==
            ➤$(placeheld[i]).attr('placeholder')) {
                // if not required, set value to empty before submitting
                $(placeheld[i]).attr('value','');
            }
        }
    });
}
</script>
```

关于此脚本, 第一点需要注意的是: 我们使用的是 Modernizr¹ JavaScript 库来检查对 placeholder 属性的支持。请在附录 A 中查询关于 Modernizr 的详细信息,

¹ <http://www.modernizr.com/>

但是现在已经完全能够了解，它提供给您许多关于出现在浏览器中指定的 HTML5 和 CSS3 功能的 `true` 和 `false` 属性。在本例中，我们使用的属性，其意义相当明显。如果浏览器支持 `placeholder`，那么 `Modernizr.input.placeholder` 将为 `true`，否则是 `false`。

如果我们确定缺乏对占位符的支持，我们可以使用 `placeholder` 属性抓取页面所有的 `input` 和 `textarea` 元素。然后对于每一个进行检测，如果该值不为空，用 `placeholder` 属性值替代该值。在此过程中，我们为元素添加 `placeholder` 类，然后您可以使 CSS 中的字体颜色变亮，或者使其看上去更像本机占位符。当用户使用伪占位符聚焦输入时，脚本清除该值并删除该 `class`。当用移除焦点时，脚本将会检测是否有值。如果没有，将添加设定的占位符文本和 `class`。

这是一个对 HTML5 补充支持的好示例：我们使用 JavaScript 对缺乏支持的浏览器提供支持，我们通过充分利用已有的 HTML5 元素和属性，而不是在 JavaScript 中借助于其他的类或硬编码值。

4.2.3 pattern 属性

`pattern` 属性使您能够提供一种正则表达式，使用户的输入与之匹配才能视为有效。对于任何 `input` 元素，用户可以输入自由格式的文本，您可以使用 `pattern` 属性来界定可接受的语法。

在模式中使用的正则表达式语言是与 JavaScript 一样的基于 Perl 的正则表达式语法，但 `pattern` 属性必须与整个值匹配，而不仅仅是一个子集。由于浏览器当前以类似于工具提示条的形式显示 `title` 属性的值，它包含比占位符文本更详细的模式指令，并形成了一个连贯的指令，所以在包含 `pattern` 属性时，您应向用户表明所期望的（要求的）模式。



关于正则表达式的简介

正则表达式是大多编程语言的一个功能，它允许开发人员指定字符模式并检查指定的字符串是否匹配该模式。正则表达式对外行来说是非常难懂的。例如，一个检查字符串是否符合 email 地址格式的正则表达式可能看起来是这样的：
`[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}`。

关于正则表达式语法的完整教程已经超出了本书的范围，如果您想了解，网上有很多的资源和相关教程。另外，您也可以从网络中进行搜索或到论坛查询，查找您想要的模式。

下面是一个简单的示例，让我们在表单的密码字段添加一个 `pattern` 属性。我们的强制要求是密码至少是 6 位字符，并且不得有空格：

```
register.html (excerpt)

<li>
  <label for="password">I would like my password to be:</label>
  <p>(at least 6 characters, no spaces)</p>
  <input type="password" id="password" name="password" required
    ➤pattern="\S{6,}">
</li>
```

`\s` 指的是“任何非空白字符，”`{6,}` 指的是“至少 6 次。”如果您想规定字符的最大数量，语法是 `\s{6,10}` 指的是 6~10 个字符。

与 `required` 属性一样，如果模式不匹配，`pattern` 属性将拒绝提交，并提供错误信息。

如果您使用的模式不是有效的正则表达式，将不会被验证。还要注意，与 `placeholder` 和 `required` 属性类似，对于不支持该属性的浏览器，您可以使用此属性的值为 JavaScript 验证代码提供一个依据。

4.2.4 disabled 属性

布尔型 `disabled` 属性存在的时间要比 HTML5 还要长，但在某种程度上已经对它进行了扩展。它可应用于除新的 `output` 元素外的任何表单控制元素——并不像 HTML 的早期版本，HTML5 允许您在 `fieldset` 上设置 `disabled` 属性，并将其应用于包含在 `fieldset` 的所有表单元素。

通常情况下，表单元素的 `disabled` 属性使在浏览器中的内容变灰——文字比启用的表单控件值的颜色要浅一些。浏览器将禁止用户在设置了 `disabled` 属性表单控件上对准焦点。此属性经常用于提交按钮的禁用，直到所有字段填写正确。例如，您可以在 CSS 中使用 `:disabled` 伪类设置禁用的表单控件的样式。

使用 `disabled` 属性的表单控件并不随表单提交，所以它们的值对服务器端的表单处理代码不可用。如果您想使该值不被用户编辑，但是能够看到并提交它，可使用 `readonly` 属性。

4.2.5 readonly 属性

`readonly` 属性类似于 `disable` 属性：它使用户不能够编辑表单字段。但是，

与 disabled 属性不同, 该字段只能够接受焦点, 其值与表单一起提交。

在评论表单中, 我们也许想包含当前页面的 URL 或评论的文章标题, 让用户知道我们正在收集此数据, 并且不允许用户对其进行更改。

```
<label for="about">Article Title</label>
<input type="text" name="about" id="about" readonly>
```

4.2.6 multiple 属性

如果使用 multiple 属性, 就表明在表单控件中可以输入多个值。虽然在 HTML 以前的版本中已经有此属性, 但它仅应用于 select 元素。在 HTML5 中, 它也可以被添加到 email 和 file 输入类型中。如果使用它, 用户可以选择多个文件, 或包含多个逗号分隔的电子邮件地址。

在编写本书时, 只在 Chrome、Opera 和 Firefox 中支持多个文件输入。



空格或逗号?

您可能注意到了用于电子邮件输入的 iOS 触摸键盘包含了一个空格。当然, 在电子邮件地址中不允许有空格。但有些浏览器允许您用空格分隔多个电子邮件地址。Firefox 4 和 Opera 都支持用逗号或空格分隔多个电子邮件地址, WebKit 不支持空格分隔, 即使空格被包含在触摸键盘中。

很快, 所有的浏览器将会支持这种额外的空格。这是大多数用户将很可能采用的输入数据方式。另外, 最近已经允许将它添加到规范中。

4.2.7 form 属性

为了不与 form 元素混淆, 在 HTML5 中的 form 属性允许您使表单元素与没有被嵌套的表单相关联。这意味着您现在可以使一个表单元件的群组或表单控件与文档中的任何其他表单相关联。form 属性将 form 元素的 id 作为其值, 与表单元件的群组或控件相关联。

如果属性被省略, 那么控件将与其嵌套的 form 一起提交。

4.2.8 autocomplete 属性

autocomplete 属性指定不管是表单还是表单控件，都应有一个自动完成的功能。对于大多数表单字段，当用户开始输入时，将出现一个下拉列表。对于密码字段，它具有在浏览器中保存密码的功能。在浏览器中支持这种功能已经多年，尽管直到 HTML5 才将它写入规范中。

在默认状态下，autocomplete 属性是开的状态。在您意识到这是最后一次所填的表单时，为了禁用它，可以使用 autocomplete="off"。这是处理敏感信息的好方法，比如信用卡号码或不需要重新使用的信息，如 CAPTCHA。

自动完成也由浏览器控制。用户可以在他们的浏览器中打开自动完成的功能。然而，如果您想覆盖这个优先选择设置，可将 autocomplete 属性设置为 off。

4.2.9 datalist 元素和 list 属性

目前仅有 Firefox 和 Opera 支持数据列表控件，但是它们确实非常酷。它们能够满足一个普通的要求：具有一组预定义自动完成选项的文本字段。和 select 元素不一样，用户可输入自己喜欢的任何数据，但当输入时，在下拉列表中会有一组建议选项呈现在用户面前。

datalist 元素，与 select 元素非常相似，是一个选项列表，每一个选项都放置在 option 元素中。然后您可以使用 input 元素的 list 属性将 datalist 与一个输入相关联。list 属性将与输入相关联的 datalist 的 id 属性作为其值。一个 datalist 可与若干个输入字段相关联。

下面是一个实际示例：

```
<label for="favcolor">Favorite Color</label>
<input type="text" list="colors" id="favcolor" name="favcolor">

<datalist id="colors">
  <option value="Blue">
  <option value="Green">
  <option value="Pink">
  <option value="Purple">
</datalist>
```

在提供支持的浏览器中，在对准焦点时，将显示一个简单的文本字段，并有一

个含有参考答案的下拉列表。图 4.4 显示了其外观。

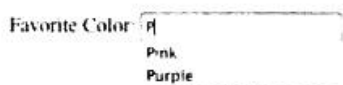


图 4.4 在 Firefox 中运行 datalist 元素

4.2.10 autofocus 属性

布尔型 autofocus 属性指定在页面加载完成时，表单控件应被对准焦点。在一个指定页面只可以有一个表单元素具有 autofocus 属性。

4.3 HTML5 新表单输入类型

您可能已经熟悉了 input 元素的 type 属性。这是一个能够决定何种表单输入将呈现给用户的属性。如果它被省略——或者是在旧版浏览器中使用新的输入类型，尽管浏览器不能识别，但仍可运行：input 元素将默认为 type="text"。这就是如今 HTML5 表单便于使用的关键。如果您使用新的输入类型，如 email 或 search，旧版浏览器将简单地呈现给用户一个标准的文本字段。

我们的注册表单目前使用您比较熟悉的 10 种输入类型中的 4 种：checkbox、text、password 和 submit。在 HTML5 之前，所有类型的完整列表如下所示。

- button。
- checkbox。
- file。
- hidden。
- image。
- password。
- radio。
- reset。
- submit。

■ text。

HTML5 带给我们的输入类型提供了更多的数据特定相关的用户界面元素以及自带的数据验证。HTML5 共有 13 种新输入类型。

■ search。

■ email。

■ url。

■ tel。

■ datetime。

■ date。

■ month。

■ week。

■ time。

■ datetime-local。

■ number。

■ range。

■ color。

让我们详细看一下这些新类型，并学习如何使用它们。

4.3.1 search

search 输入类型（`type="search"`）提供一个搜索字段——一个一行的文本输入控件，可以输入一个或多个搜索术语。规范说明如下：

文本状态与搜索状态的区别主要是在格式上：搜索字段平台与常规文本字段平台的不同之处在于，搜索状态可能导致外观与平台搜索字段一致，而不像一个常规的文本字段。

许多浏览器在设置 search 输入的样式时，与浏览器或操作系统搜索框风格一致。一些浏览器添加了单击鼠标清除输入的功能，一旦文本输入到字段中，便提供

一个 x 图标。图 4.5 显示的是在 Mac OS X 操作系统的 Chrome 浏览器中的运行状况。

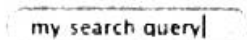


图 4.5 将 search 输入类型的样式设置为与操作系统的搜索字段相似

目前,只有 Chrome 和 Safari 提供按钮来清除字段。Opera 11 显示一个圆角框,没有控件来清除字段,但如果采用了任何样式,比如背景颜色,便切换显示为正常的文本字段。

对于搜索字段,您仍可以使用 `type="text"`,新的搜索类型为用户提供视觉提示,使他们到需要去的地方搜索网站,并提供一个用户习惯的界面。The THML5 Herald 没有搜索字段,但这里有一个如何使用它的示例:

```
<form id="search" method="get">
  <input type="search" id="s" name="s">
  <input type="submit" value="Search">
</form>
```

与其他新的输入类型一样,在不支持 search 的浏览器中使用它,它显示为一个常规文本框,在适当的时候,我们没有理由不使用它。

4.3.2 Email Addresses

email 类型 (`type="email"`),毋庸置疑,用于指定一个或多个电子邮件地址。它支持布尔型 `multiple` 属性,允许多个、逗号分隔的电子邮件地址。

让我们使用 `type="email"` 对表单注册人的电子邮件地址进行更改:

register.html (excerpt)

```
<label for="email">My email address is</label>
<input type="email" id="email" name="email">
```

如果您想将输入类型从 text 更改为 email,正如我们现在这里所做的,您将会注意到用户界面没有视觉变化:输入元素仍看起来像纯文本字段。然而,在屏幕后面却存在不同。

如果您使用 iOS 设备,变化就会很明显。当您聚焦电子邮件字段时,iPhone、iPad 和 iPod 都会显示一个优化电子邮件输入的键盘(有一个 @ 符号的快捷键),如图 4.6 所示。



图 4.6 在 iOS 设备上, email 输入类型提供一个特殊键盘

Firefox、Chrome 和 Opera 还提供电子邮件输入错误信息:如果您尝试提交的表

单含有无法识别的内容，其内容形式为一个或多个电子邮件地址，浏览器就会告诉您是什么错误。默认的错误信息如图 4.7 所示。

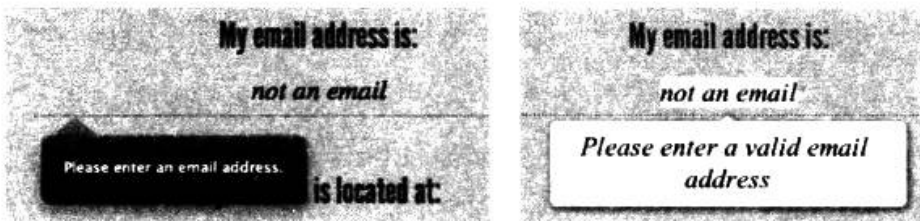


图 4.7 在 Firefox 4 (左图) 和 Opera 11 (右图) 中，关于不正确的电子邮件地址格式的错误信息



自定义验证信息

不喜欢提供的错误信息吗？在某些浏览器中，您可以使用 `.setCustomValidity(errorMessage)` 设置您自己的错误信息。`.setCustomValidity` 仅将您想提供的错误信息当作唯一的参数。如果您想完全删除错误信息，可向 `.setCustomValidity` 传递空字符串。

不幸的是，尽管您可以改变信息的内容，但您不能更改它的外观，至少目前是这样。

4.3.3 URL

`url` 输入类型 (`type="url"`) 是用于说明网站地址的。与 `email` 非常相像，显示为一个常规的文本字段。在许多触摸屏中，屏幕键盘为网站地址的输入进行了优化，具有正斜杠 (/) 和 “.com” 快捷键。

让我们使用 `url` 输入类型更新我们的注册表：

register.html (excerpt)

```
<label for="url">My website is located at:</label>
<input type="url" id="url" name="url">
```

Opera、Firefox 和 WebKit 支持 `url` 输入类型，如果 URL 格式不正确，将会报告输入无效。只有常规的 URL 格式才能通过验证。因此，例如，`q://example.xyz` 被认为是有效的，即使 `q://` 不是一个真正的协议以及 `.xyz` 不是一个真正的顶级网域名称。同样，如果您想使输入的值符合更具体的格式，请在标签（或占位符）提供信息以便使用户了解，并且可以使用 `pattern` 属性以确保正确，我们将在本章后

面的章节详细介绍 pattern 属性。



WebKit

我们在本书中所提到的 WebKit，是指使用 WebKit 呈现引擎的浏览器，包括 Safari（包括在桌面上以及 iOS 上）、Google Chrome、Android 浏览器以及许多其他移动浏览器。您可以在 <http://www.webkit.org> 网站上找到关于 WebKit 开源项目的更多信息。

4.3.4 Telephone Numbers

对于电话号码，使用 tel 输入类型（type="tel"）。与 url 和 email 类型不同，tel 类型不会强制执行特定的语法或模式。字母和数字——只要是不是新的一行或回车符的其他任何字符都是有效的。这里有一个很好的理由：世界上各个国家有许多不同类型的有效电话号码，并具有不同的长度和标点符号，所以不可能指定唯一的格式作为标准。例如，在美国，+1(415)555-1212 通常被理解为 415.555.1212。

您可以通过包含使用正确语法的占位符支持一种特殊格式，或在输入类型后用一个示例加以注释。另外，您可以通过使用 pattern 属性规定一种格式或使用 setCustomValidity 方法提供客户端验证。

4.3.5 Numbers

number 类型（type="number"）提供了一个输入数字的输入类型。通常，这是一个“微调”框，在这里您可以输入一个数字或单击向上或向下的箭头来选择一个数字。

让我们使用 number 输入类型来更改数量字段：

register.html (excerpt)

```
<label for="quantity">I would like to receive <input type="number"
name="quantity" id="quantity"> copies of <cite>The HTML5 Herald
</cite></label>
```

图 4.8 显示了在 Opera 浏览器中的运行结果。

I would like to receive 3 copies of The HTML5 Herald

图 4.8 在 Opera 浏览器中所见到的 number 输入类型

数字输入类型由 `min` 和 `max` 属性来规定最大和最小允许值。我们强烈推荐您使用这些属性，否则使用向上和向下箭头可能会导致不同的值（非常奇怪），这取决于浏览器。



什么时候一个号码不是数字？

也许会有很多时候，当您想使用 `number` 类型的时候，其实另一种输入类型更合适。例如，街道地址是一个数字似乎更为合理。但是想想看：您是否想单击微调框直到 34154？更重要的是，许多街道号码有非零部分：试想 24½ 或 36B，这两个都不适合用数字输入类型。

此外，账户号码可以是字母和数字的混合，或有破折号。如果您知道号码的模式，请使用 `pattern` 属性。如果范围广泛，或者号码包含非数字字符并且必须使用字段的时候，请记住不要使用 `number` 类型。如果字段是可选的，为了使数字键盘成为触摸屏设备的默认设定，您也许会无论如何都想使用 `number` 类型。

如果您决定采用 `number` 输入类型的方式，请记住在 `number` 类型中是不支持 `pattern` 属性的。换句话说，如果浏览器支持 `number` 类型，那么它将取代任何 `pattern`。也就是说，可以任意包含 `pattern`，在浏览器支持 `pattern` 的情况下，便不会支持 `number` 输入类型。

您也可以提供 `step` 属性，这样可以通过单击上和下的箭头使数值增加或减少。在 Opera 和 WebKit 中，都支持 `min`、`max` 以及 `step` 属性。

在一些触摸屏设备上，将焦点对准到 `number` 输入类型，将会出现一个数字触摸板（而不是一个完整的键盘）。

4.3.6 Ranges

`range` 输入类型 (`type="range"`) 在支持该输入类型的浏览器（目前的 Opera 和 WebKit）中显示一个滚动控件。和数字类型一样，它允许使用 `min`、`max` 和 `step` 属性。根据规范，`number` 和 `range` 的区别是：在 `range` 类型中所输入的具体数的精确值是不重要的。如果您想得到一个不精确的数字，这是一个理想的输入类型。例如，一个顾客满意度调查，可以让客户为他们所接受到的服务进行评级。

让我们使用 `range` 输入类型改进我们的注册表单。字段让用户对他们所掌握的 HTML5 知识进行评级，等级从 1 到 10。

register.html (excerpt)

```
<label for="rating">On a scale of 1 to 10, my knowledge of HTML5
is:</label>
<input type="range" min="1" max="10" name="rating" type="range">
```

step 属性的默认值是 1，所以它不是必需的。图 4.9 显示了在 Safari 中的运行结果。

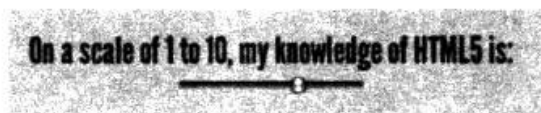


图 4.9 在 Chrome 浏览器中显示的 range 输入类型

range 输入类型的默认值是滚轴的中间点——换句话说，就是最小值和最大值之间的中间值。

规范规定：如果指定的最大值小于最小值，就允许使用反向滚动轴（数值从右到左，而不是从左到右）。然而，目前没有浏览器支持此技术。

4.3.7 Colors

color 输入类型 (type="color") 为用户提供了颜色选取器——至少它能在 Opera 浏览器中运行（并且，令人吃惊的是，在比较新的黑莓智能手机的内置浏览器中也可以使用）。颜色选取器会返回一个十六进制的 RGB 颜色值，比如 #FF3300。

如果您想使用 color 输入提供占位符文本，表明必须使用十六进制 RGB 颜色格式，并使用 pattern 属性对其进行限制，只允许输入有效的十六进制颜色值，那么就必须等到完全支持这个输入类型。

我们不在表单中使用 color 输入类型，但是，如果采用此输入类型，应使用下列代码：

```
<label for="clr">Color: </label>
<input id="clr" name="clr" type="text" placeholder="#FFFFFF"
pattern="#(?:[0-9A-Fa-f]{6}|[0-9A-Fa-f]{3})" required>
```

由此代码生成的颜色选取器如图 4.10 所示。单击 **other...** 按钮，将会呈现一个完整的颜色盘，使用户选择任何十六进制的颜色值。

WebKit 浏览器也支持颜色输入类型，并能够显示颜色是否有效，但是还不能提

供颜色选取器。

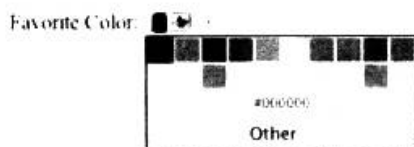


图 4.10 Opera 中针对 color 输入类型的颜色选取器

4.3.8 Dates and Times

现在有一些新的日期和时间输入类型，包括 `date`、`datetime`、`datetime-local`、`month`、`time` 和 `week`。所有日期和时间输入将接受符合 ISO9601 标准的数据格式¹。

Date

它包括日期（年，月，日），但不包括时间。例如，2004-06-24。

Month

仅包括年和月。例如，2012-12。

Week

它包括年和周数（从 1 到 52）。例如，2011-W01 或 2012-W52。

Time

一天的时间，使用军用格式（以 24 小时记时）。例如，22:00 代替 10.00pm。

Datetime

它包含日期和时间，由“T”分隔，在后面跟一个“Z”代表 UTC（Coordinated Universal Time，世界标准时间），或由 `a+` 或 `-` 字符指定的时区。例如，“2010-03-17T10:45-5:00”表示 2011 年 3 月 17 日上午 10 点 45 分，UTC 减去 5 个小时时区（美国东部标准时间）。

Datetime-local

除了省略了时区，其他都与 `datetime` 一样。

¹ http://en.wikipedia.org/wiki/ISO_8601

这些类型中最常用的是 date。规范要求浏览器显示日期控件，在编写本书时，仅 Opera 通过提供日历控件支持此技术。

让我们使用 date 输入类型改进我们的认购开始日期字段：

register.html (excerpt)

```
<label for="startdate">Please start my subscription on:</label>
<input type="date" min="1904-03-17" max="1904-05-17"
  id="startdate" name="startdate" required aria-required="true"
  placeholder="1911-03-17">
```

现在，当我们通过 Opera 浏览器查看表单时，将会有有一个日历控件，如图 4.11 所示。不幸的是，目前还不能使用 CSS 设置样式。

对于 month 和 week 类型，Opera 显示同一个日期选择器，但只允许用户选择整月或整星期。在这种情况下，无法选择单独某一天；相反，单击某一天时，选择的是整月和整周。

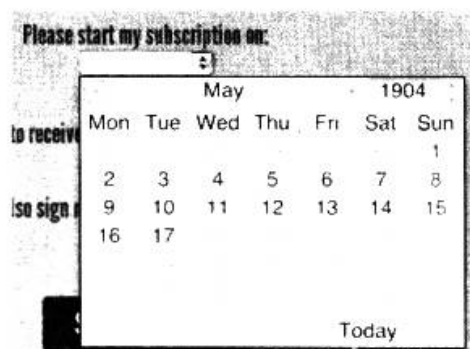


图 4.11 Opera 针对 date、datetime、datetime-local、week 和 month 输入类型的日期选择器

目前，WebKit 对 date 输入类型提供一些支持，为用户提供一个类似于 number 类型的界面，并有一个向上和向下的箭头。在 Safari 浏览器上使用该控件，运行起来会有一些奇怪，默认值是公历的第一天：1582-10-15。在 Chrome 浏览器中默认值是 0001-01-01，并且最大值是 275760-09-13；而 Opera 的默认值是当天日期。由于这些比较奇怪，因此在使用任何基于日期的输入类型时，我们强烈建议包含最小值和最大值（上述除 time 外的所有输入类型）。就像使用 number 时，与它的 min 和 max 属性一同使用。

刚才添加到开始日期字段的 placeholder 属性，在 Opera 的日期选择器界面中是多余的，但是将它留在原处指导其他浏览器的用户，还是非常合理的。

最后，当所有浏览器支持所有新输入类型的 UI 时，placeholder 属性将只与 text、search、URL、telephone、email 和 password 类型相关。直到那时，占位符是一个很好的方式，提示用户在那些字段中应填入何种数据——请记住，在不支持的浏览器中，它们看起来仅是一些常规的文本字段。



动态日期

在我们上面的示例中，我们将 min 和 max 值硬编码到 HTML 中。例如，如果您想将最小值设置为当前日期的第二天（将它设定为报纸订购起始日期更具实际意义），这样就需要每天更新 HTML。最好的办法是在服务器端动态生成允许范围内日期的最小值和最大值。可以用一小段 PHP 代码达到此目的：

```
<?php
function daysFromNow($days){
    $added = ($days * 24 * 3600) + time();
    echo(date("Y-m-d", $added));
}
?>
```

在我们原来的标记中是静态日期，现在用上面函数动态生成日期：

```
<li>
    <label for="startdate">Please start my subscription on:
    </label>
    <input type="date" min="<?php daysFromNow(1); ?>"
    max="<?php daysFromNow(60); ?>" id="startdate"
    name="startdate" required aria-required="true"
    placeholder="1911-03-17">
</li>
```

这样，限制用户输入的信息会使表单内容更合情合理。

您也可以在使用 date 和 time 输入类型时包含 step 属性。例如，将 month 的 step 属性设置为 step="6"，这样就限制用户只能选择一月或七月。设置 time 和 datetime 输入类型时，step 属性必须用秒来表示，因此，在设置 time 输入类型时，step="900"会使输入以 15 分钟为增量进行递增。

4.4 HTML5 中的其他新表单控件

我们已经介绍了新 input 元素的 type 属性值，以及一些在大多数表单元素中都有效的属性，但是 HTML5 的 Web 表单仍为我们提供了更多内容。在 HTML5 中，有 4 个新的表单元素：output、keygen、progress 和 meter。由于 progress 和 meter 经常用于表单外部，因此我们将在最后一章介绍它们。现在让我们先看看另外两个元素。

4.4.1 output 元素

output 元素的目的是接受并显示计算的结果。在用户可以看到值但不能直接操作它时，以及能够从输入表单的其他值计算得出该值时，将使用 output 元素。例如，购物车中的总成本是计算运输和所交税以后的值。

output 元素的值被包含在开始和结束标记之间。一般来说，在浏览器中使用 JavaScript 更新此值更为合理。output 元素有一个 for 属性，它用于引用表单字段的 id，而这些表单字段中的值将用于计算 output 元素的值。

值得注意的是，output 元素的名称和值与表单一起提交。

4.4.2 keygen 元素

keygen 元素是生成公共-私有密钥对¹，并提交密钥对中公共密钥的控件。Opera、WebKit 和 Firefox 都支持此元素，尽管提供不同的选项，但都将它呈现为一个含有所生成密钥长度选项的下拉菜单。

keygen 元素引入了两个新属性：challenge 属性指定与公共密钥一起提交的字符串，keytype 属性指定所生成密钥的类型。在编写本书时，仅有 rsa 支持 keytype，rsa 是用于公共密钥加密的一种普通算法。

4.5 对现有表单控件及属性的更改

在 HTML5 中对表单控件进行了一些其他更改。

4.5.1 form 元素

在本章中，我们一直在谈论应用于各种表单字段元素的属性；然而，form 元素自身也有一些特有的新属性。

首先，正如我们看到的，HTML5 对本地验证表单字段提供了许多方式；某些输入类型，如 email 和 url，以及 required 和 pattern 属性。然而，由于样式设置和语义的原因，您可能想在使用这些输入类型和属性时，取消对表单提交的

¹ http://en.wikipedia.org/wiki/Public-key_cryptography

阻止。新的布尔型 `novalidate` 属性允许表单提交而无需本地验证其字段。

接下来，表单不再需要定义的 `action` 属性。如果省略它，表单将默认为 `action` 属性设置为当前页。

最后，我们可以将先前介绍的 `autocomplete` 属性直接添加到 `form` 元素中。在这种情况下，它将应用于表单的所有字段，除非这些字段用自身的 `autocomplete` 属性覆盖了它。

4.5.2 optgroup 元素

在 HTML5 中，可以将 `optgroup` 当作另一个 `optgroup` 的子代，这对于多层选择菜单来说很有用。

4.5.3 textarea 元素

在 HTML4 中，我们需要通过定义 `rows` 和 `cols` 属性的值来指定 `textarea` 元素的大小。在 HTML5 中，不再需要这些属性，您应使用 CSS 定义 `textarea` 的宽度和高度。

在 HTML5 中新增的是 `wrap` 属性。这个属性应用于 `textarea` 元素，并可以有 `soft`（默认值）或 `hard` 值。使用 `soft` 值时，提交的文本没有用户实际输入的断行，而在使用 `hard` 值时，将会提交浏览器由于字段大小而生成的任何断行。若您将 `wrap` 设置为 `hard`，则需要指定 `cols` 属性。

4.6 小结

由于对 HTML5 输入元素和属性的支持力度不断增强，因此网站将会越来越少地使用 JavaScript 来进行客户端验证和用户页面改进，而由浏览器处理大多数繁重的工作。传统的用户代理可能还在等待可预见的未来，但是没有理由停止前进，应恰当地使用补充支持及后备技术填补需要的空白，从而更好地使用 HTML5 Web 表单。

HTML5 音频和视频

如果没有涉及新的 `video` 和 `audio` 元素,那么本书就不能称之为关于 HTML5 完整的一本书。这些突破性的新元素已经被应用于 Web,虽然性能有限,但是越来越多的开发人员和内容提供商开始将它们纳入项目之中。

对于 The HTML5 Herald,我们将视频元素放在 3 个布局栏中的第一栏。在我们探讨 `video` 元素、它的各种属性及其相关元素的详细内容之前,让我们先看一下当前在 Web 上的视频状况。

在大多数情况下,本章将重点介绍 `video` 元素,因为在我们的示例项目中将要使用它。然而,`audio` 元素与之基本相同:几乎所有用于视频的属性和功能也同样应用于音频。当然也有一些例外,我们一定会指出来。

5.1 历史简介

到目前为止,在大多数情况下,是通过第三方插件或集成在 Web 浏览器上的应用程序将在 Web 上的多媒体内容放到网页上。例如,此类软件包括 QuickTime、RealPlayer 和 Silverlight。

目前最流行的方式是通过 Adobe 的 Flash Player 插件将视频和音频嵌入到网页中。Flash Player 插件最初由 Macromedia 开发,由于 2005 年 Adobe 买断了该公司,

因此现在由 Adobe 公司对其进行维护。该插件从 20 世纪 90 年代中期以来,便已经可以使用,但直到 21 世纪才开始作为服务于视频内容的一种方式。

在 HTML5 之前,没有将视频嵌入到网页的标准方式。像 Adobe 的 Flash Player 之类的插件完全由 Adobe 控制,并不向共享开发开放。

在 HTML5 中引入 video 和 audio 元素解决了此问题,并使多媒体成为网页的无缝组成部分,就像 img 元素。使用 HTML5,不需要用户下载第三方软件来观看您的内容,并且视频和音频播放器更容易通过脚本访问。

5.2 目前状况

不幸的是,HTML5 的视频和音频仅在理论上是非常出众的,在实际应用中并没有那么简单。在您决定在页面中包含 HTML5 的多媒体新元素时,需要考虑很多因素。

首先,您需要考虑的是浏览器的支持状态。在编写本书时,占有非常大的市场份额的 Internet Explorer 8 和更早版本的 IE 浏览器并不支持 HTML5 自带的视频和音频。不幸的是,现在仍有很多用户使用 IE 浏览器。

其他浏览器制造商对 HTML5 视频提供了广泛支持,现在已经得到广泛应用(Chrome 3 及更高版本、Safari 4 及更高版本和 Firefox 3.5 及更高版本)。不支持 HTML5 的 Chrome 浏览器的最后一个版本(第 2 版)已经没有市场份额了,不支持 HTML5 视频的 Safari 和 Opera 版本的浏览器也同样如此。

尽管 Internet Explorer 所占的市场份额很大,现在您仍可以在网页上使用 HTML5 视频。稍后,我们将向您演示如何将新视频元素设计为向后兼容的,这样在使用不支持的浏览器时,将仍可以访问多媒体内容。

5.2.1 视频容器格式

网络上的视频是基于容器格式和编解码器的。容器是一个储存由可访问的视频文件所构成的所有必要数据的包装器。很像一个 ZIP 文件包装或包含着文件。例如,一些非常有名的视频容器,包括 Flash Video (.flv)、MPEG-4 (.mp4 或.m4v)和 AVI (.avi)。

视频容器储存的数据包括视频轨道、帮助音频视频同步的音轨标记、语言信息以及其他一些描述内容的元数据。

与 HTML5 相关的视频容器格式是 MPEG-4、Ogg 和 WebM。

5.2.2 视频编解码器

视频编解码器定义了多媒体数据流编码和解码的算法。编解码器可以对数据流进行编码，使之用于传输、存储或加密，或可以对其解码进行回放或编辑。为了达到使用 HTML5 视频的目的，我们关注的是对数据流的解码以及回放。使用最多的 HTML5 视频解码文件是 H.264、Theora 和 VP8。

5.2.3 音频编解码器

音频编解码器与视频解码文件工作理论上是一样的，音频播放器主要涉及声流，而不是视频帧。使用最多的音频编解码器是 AAC 和 Vorbis。

5.2.4 当前浏览器使用哪种组合

如果浏览器的技术支持允许我们选择一个容器、视频编解码器、音频编解码器创建一个使用 HTML5 的新 video 元素嵌套视频的标准方式，那将是非常好的一件事情。不幸的是，事实上并没有那么简单，尽管现在正在逐步改善。

在表 5.1 中，我们描述了在使用最广泛的浏览器版本中所支持的视频容器和编解码器。

表 5.1 对 HTML5 视频的浏览器支持

| 容器/视频编解码器/音频编解码器 | Firefox | Chrome | IE | Opera | Safari | iOS Safari | Android |
|-------------------|-----------|---------|----------------------|------------|---------|------------|------------------------|
| Ogg/Theora/Vorbis | 3.5 及更高版本 | 3 及更高版本 | - | 10.5 及更高版本 | - | - | - |
| MP4/H.264/AAC | - | 3-11 | 9 及更高版本 | - | 4 及更高版本 | 4 及更高版本 | 2.1 及更高版本 ^a |
| WebM/VP8/Vorbis | 4 及更高版本 | 6 及更高版本 | 9 及更高版本 ^b | 10.6 及更高版本 | - | - | 2.3 及更高版本 |

^a 在 2.3 之前的 Android 版本需要 JavaScript 播放视频。

^b 用户在 Windows 上安装 VP8 编解码器后，IE9 支持用 VP8 回放 WebM 的视频。

Opera Mini 和 Opera Mobile 目前没有提供对 HTML5 的支持，但是 Opera 已经声明在以后的发布¹中计划引入该技术支持。

¹ <http://my.opera.com/operamobile/blog/2010/12/04/developing-opera-mobile-for-android/>



许可证问题

虽然可以在任何情况下免费使用新的 video 元素，但容器和编解码器并不那么简单。例如，虽然 Theora 和 VP8（WebM）编解码器没有遇到专利阻碍，但是 H.264 编解码器遇到了专利阻碍，须由 MPEG-LA 集团授权。

目前对于 H.264，如果免费向用户提供视频，那么您不需要支付版税。然而，详细的许可证问题远远超出本书的范畴和意图，所以需要小心，在页面添加 HTML5 视频时、使用任何特定视频格式之前，尽可能多做一些调查。

5.3 标记

在介绍关于容器、编解码器、浏览器技术支持以及许可证问题的商业信息后，现在我们需要探讨一下 video 元素的标记及其相关属性。

在网页上包含 HTML5 视频的最简便方法如下：

```
<video src="example.webm"></video>
```

但是，从前几章的介绍中，您可能已经看出，这样的代码只能够在有限的几个浏览器中运行。但是，在某种程度上，它是运行 HTML5 视频所需的最基本代码。如果是更完美的话，它应在任何地方都可以运行——就像 img 元素可在任何地方运行一样——但是达到这种程度，还尚有一些距离。

与 img 元素相似，video 元素也应包含 width 和 height 属性：

```
<video src="example.webm" width="375" height="280"></video>
```

即使在标记中设置它的尺寸，也并不影响视频的高宽比。例如，上面示例中的视频实际上是 375 像素×240 像素，并且标记如上所示，但视频将在 HTML 指定的 280 万像素的空间内垂直居中。这防止了视频不必要的拉伸和扭曲。

width 和 height 属性只接受整数，它们的值以像素为单位。当然，这些值可以通过脚本或 CSS 覆盖。

5.3.1 启用本机控件

如果不能使用户播放、暂停、停止、搜索整个视频，或调节音量，都不能称之

为完整的嵌套视频。HTML5 的 video 元素包含了 controls 属性来实现这些功能：

```
<video src="example.webm" width="375" height="280" controls></video>
```

Controls 是一个布尔属性，所以不需要任何值。在标记中包含它，相当于告诉浏览器使控件可见并且用户可以访问。

每一个浏览器负责内置视频控件的外观。图 5.1~图 5.4 显示了在不同的浏览器中，这些控件不同的外观。



图 5.1 在 Firefox 4 浏览器中的本机视频控件



图 5.2 在 Internet Explorer 9 中的本机视频控件



图 5.3 在 Opera 11 中的本机视频控件



图 5.4 在 Chrome 中的本机视频控件

5.3.2 autoplay 属性

由于在许多情况下并不使用这个属性，因此我们很想省略对这个特定属性的引用。然而，在有些情况下，还是比较恰当的。布尔 autoplay 属性正如它的字面意思：它告诉网页尽快播放视频。

通常情况下，这不是一个好的做法。我们深知，当页面加载完毕，马上播放视频或音频是非常令人不愉快的——尤其是在音箱处于打开的状态。可用的最好做法是规定网页上任何声音及运行都应在用户发出请求时启动。但是这并不意味着 autoplay 属性没有用处。

例如，发生问题的网页只包含视频而没有其他内容——也就是说，用户单击网页的链接的唯一目的就是观看特定的视频——自动播放是可以接收的，当然还取决于视频的大小、周边内容及用户。

下面是实现此功能的代码：

```
<video src="example.webm" width="375" height="280" controls
➡autoplay></video>
```



在 iPhone 上的自动播放

在 iPhone 上的 Safari 浏览器将忽略 `autoplay` 属性; 所有视频在开始前都要等到用户按播放按钮。这是非常明智的, 因为手机通常有带宽限制。

5.3.3 loop 属性

在使用另一个布尔 `loop` 属性前, 您应该再三考虑。相当明显: 根据规范, 在使用此属性时, 将告诉浏览器播放到结尾处时, 应再回到媒体资源的开始处。

所以, 您创建一个网页的唯一目的是为了惹恼访问者, 可以包含以下代码:

```
<video src="example.webm" width="375" height="280" controls
➡autoplay loop></video>
```

自动播放并且无限循环! 我们只需删除这个本机控件并取得一箭三雕的效果。

当然, 也有一些情况下使用 `loop` 属性: 设想一个基于浏览器的游戏, 只要网页打开, 背景声音及音乐就应连续播放。

5.3.4 preload 属性

与前两个属性相比, `preload` 属性在许多情况下都很有用。Preload 属性可以接受下面 3 个值中的一个。

auto

`auto` 值表明视频和它相关的元数据在视频播放前开始加载。这样, 在用户发送请求时, 浏览器可以更快地开始播放视频。

none

`none` 值表明在用户按播放按钮之前, 视频将不在后台进行加载。

metadata

它与 `none` 值非常相像, 只有一点除外: 即使没有加载视频本身, 也预装载与

视频相关的任何元数据（如它的尺寸、持续时间等）。

这个特殊属性没有规范定义的默认值，在一些情况下，通常被省略：每个浏览器都可以设定三个值中的一个为默认状态，这样更合情合理。它允许桌面浏览器自动预装载视频和/或元数据（确实没有不良效果），同时也允许手机浏览器默认为元数据或空，这是因为许多手机用户受到带宽限制，并更喜欢有下载视频与否的选择。

5.3.5 poster 属性

当您上网观看视频时，通常显示一个视频单帧来提供内容摘要。poster 属性使其更方便于选择这样一个内容摘要。这个属性类似于 src，它在服务器端通过 URL 的方式指向图像文件。

下面是如何使用 video 元素的 poster 属性定义的代码：

```
<video src="example.webm" width="375" height="280"  
➡poster="teaser.jpg" controls></video>
```

虽然 poster 属性非常有用，但是在 iOS 3（在 iOS 4 中已经更正）中有一个程序错误，如果使用该属性，那么它将阻止视频播放。若您知道您的用户使用 iOS 3.X，则应避免使用 poster 属性，或针对这些设备删除它。

5.3.6 audio 属性

audio 属性为 video 元素控制音轨的默认状态，并且目前只接受一个值：muted。规范中规定：在将来，可能会添加其他值，例如，用来指定默认音轨或音量。

muted 值将会使视频音轨默认为静音，可能覆盖任何用户的优先选择。这将仅控制元素的默认状态——用户与控件交互，或使用 JavaScript 更改它。

在此将它添加到 video 元素中：

```
<video src="example.webm" width="375" height="280"  
➡poster="teaser.jpg" audio="muted"></video>
```

5.3.7 添加对多种视频格式的支持

正如我们前面讨论过的，使用单一容器格式服务于您的视频在目前是不可行的，尽管这确实是使用 video 元素的最终构想，我们也希望在将来能够实现。为了能够

包含多种视频格式，video 元素允许定义 source 元素，这样使每一个用户代理能够用所选择的格式显示视频。这些元素实现同一种功能，就像 video 元素的 src 属性。所以如果您提供了 source 元素，便不需要为您的视频指定一个 src 元素。

考虑到当前浏览器的支持，我们可以声明 source 元素如下：

```
<source src="example.mp4" type="video/mp4">
<source src="example.webm" type="video/webm">
<source src="example.ogv" type="video/ogg">
```

source 元素（很奇怪）使用 src 属性指定视频文件的位置。它也接受 type 属性为被请求的资源规定容器格式。后一种属性允许浏览器判定是否能够播放有问题文件，这样防止浏览器不必要地下载一些不支持的格式。

type 属性也允许规定编解码器参数，此参数为被请求的文件定义了视频和音频编解码器。以下是关于 source 元素及指定的编解码器的代码：

```
<source src="example.mp4"
  type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'>
<source src="example.webm" type='video/webm; codecs="vp8, vorbis"'>
<source src="example.ogv" type='video/ogg; codecs="theora, vorbis"'>
```

您会注意到，为了与容器和编解码器的值能够相符，type 属性的语法略有改动。值的双引号已经改为单引号，而且为编解码器特别包含了另一组嵌套双引号。

乍一看上去，可能有一点点混乱，但是一旦您有了一套解码视频的方法（我们将在本章的后面接触到这方面内容），在多数情况下就只是复制和粘贴这些值。重要的一点是您为指定的文件定义了正确的值，以确保浏览器可以判定哪个（如有）文件可以播放。

5.3.8 资源顺序

在我们上面的示例中，首先包含 MP4/H.264/AAC 容器/编解码器的组合。这就确保了视频在 iPad 上能够播放。在这个设备上，错误程序将导致第一个 source 元素被识别。这样能够安全地假设这个错误程序将在 iPad 的下一个版本中得到解决，但是现在包含 MP4/H.264 文件首先是确保兼容性。

Internet Explorer 9、Safari 以及旧版 Chrome 浏览器将识别第一个 source 元素，这样便覆盖了大部分 HTML5 的准用户。

在列表中的下一个元素定义了 WebM/VP8/Vorbis 容器/编解码器组合。高版本的

Chrome 浏览器对此提供支持，并最终将放弃对 H.264 的支持。除了 Chrome，WebM 视频将在 Firefox 4 和 Opera 10.6 中播放。

最后，第三个 source 元素声明了 Ogg/Theora/Vorbis 容器/编解码器组合，Firefox 3.5 和 Opera 10.5 提供支持。虽然其他浏览器也支持这种组合，但是由于它们在资源顺序中已提前出现，因此将使用其他格式。仅支持此种组合的浏览器是旧版浏览器，它们的当前版本支持其他格式，所以将来一旦这些版本相当罕见时，便有可能放弃支持此种格式。

将这 3 个 source 元素作为 video 元素的子代，所以声明这 3 个文件格式的代码如下：

index.html (excerpt)

```
<video width="375" height="280" poster="teaser.jpg" audio="muted">
  <source src="example.mp4" type='video/mp4;
  ➡codecs="avc1.42E01E, mp4a.40.2"'>
  <source src="example.webm" type='video/webm;
  ➡codecs="vp8, vorbis"'>
  <source src="example.ogv" type='video/ogg;
  ➡codecs="theora, vorbis"'>
</video>
```

您会注意到上面的代码，在 video 元素中现在没有 src 属性。同样由于是多余的，因此它也将覆盖定义在 source 元素中的任何视频文件。

5.3.9 关于 Internet Explorer 6~8

在 video 元素中所包含的 source 元素将涵盖所有的现代浏览器，但我们仍要确保视频能够为潜在的大部分用户播放。如前所述，很大比例的用户使用的浏览器仍不能够全面支持 HTML5 视频。多数用户仍在使用 Internet Explorer 9 之前的版本。

为了遵循适当降格原则，HTML5 的 video 元素允许旧版浏览器可以通过其他方式访问视频。不能够识别 video 元素的旧版浏览器将忽略该元素以及它的子代 source 元素。但是如果浏览器能够将 video 元素所包含的内容视为有效的 HTML，那么它将读取和显示该内容。

那么什么样的内容可以服务于那些不提供支持的浏览器呢？根据 Adobe¹ 记载，

¹ http://www.adobe.com/products/player_census/flashplayer/version_penetration.html

99%的用户已经在系统上安装了 Flash 插件，另外，大多情况下 Flash 插件是第 9 版本或更高版本，这些都支持 MPEG-4 视频容器格式。为了允许 Internet Explorer 6~8（以及其他不支持 HTML5 视频的旧版浏览器）播放视频，我们声明一个嵌入的 Flash 视频作为备用。以下是关于 The HTML5 Herald 的完整代码，以及 Flash 备用代码：

index.html (excerpt)

```
<video width="375" height="280" poster="teaser.jpg" audio="muted">
  <source src="example.mp4" type='video/mp4;
  ➤codecs="avc1.42E01E, mp4a.40.2"'>
  <source src="example.webm" type='video/webm;
  ➤codecs="vp8, vorbis"'>
  <source src="example.ogv" type='video/ogg;
  ➤codecs="theora, vorbis"'>
  <!-- fallback to Flash: -->
  <object width="375" height="280" type="application/x-shockwave-
  ➤flash" data="mediaplayer-5.5/player.swf">
    <param name="movie" value="mediaplayer-5.5/player.swf">
    <param name="allowFullScreen" value="true">
    <param name="wmode" value="transparent">
    <param name="flashvars" value="controlbar=over&image=
  ➤images/teaser.jpg&file=example.mp4">
    <!-- fallback image -->
    
  </object>
</video>
```

这里虽然不深入地讨论新添加的代码是如何工作的（因为这毕竟不是一本关于 Flash 的书），但是除了标记之外，还有以下几点需要注意。

- ❶ Object 元素的 width 和 height 属性应与 video 元素中的 width 和 height 属性相同。
- ❷ 我们使用的是由 LongTail Video¹提供的开放源码 JW 播放器播放文件；您可以使用您喜欢的播放器。
- ❸ Flash 视频代码有一个自身的备用代码——如果 Flash 视频代码不能够工作，将显示一个图像文件。
- ❹ 第四个 para 元素定义了将被使用的文件（example.mp4）。如上所述，大多情况下，现在的 Flash 播放器都支持使用 MPG-4 容器格式播放视频，所以

¹ <http://www.longtailvideo.com/players/jw-flv-player/>

没有必要编码另一种视频格式。

- ❏ 规范规定，支持 HTML5 视频并启用 HTML5 的浏览器忽略任何在 video 元素（不是 source 标签）中的内容，所以备用代码在所有浏览器中都是安全的。

这里我们最后要提到的一点是，除了 Flash 后备内容，您也可以提供一个可选的“下载链接”，允许用户得到一个视频的本地副本，可以在闲暇时观看。这样确保每个人都可以通过各种方式观看视频。

5.3.10 MIME 类型

如果您发现已经严格按照我们的指示去做，但视频仍不能在服务器端播放，那么问题可能与发送的内容类型信息相关。

内容类型又称为 MIME 类型，它告诉浏览器正在查看的内容类型。这是一个文本文档？如果是，是什么类型？HTML？JavaScript？还是一个视频文件？内容类型将为浏览器回答这些问题。每一次浏览器请求页面时，服务器在发送任何文件前都会先向浏览器发送一个“header”，这些 headers 会告诉浏览器如何诠释后面即将发送过来的文件。内容类型是服务器发送给浏览器的许多 headers 之一的一个示例。

通过 source 元素包含的每个视频文件的 MIME 类型与 type 属性的值是一样的（减去所有的编解码器信息）。出于 HTML5 视频的目的，我们只关注 3 个 MIME 类型。确保服务器能够播放所有类型视频文件，请将下面几行代码放在您的 .htaccess 文件中（如果您使用的 Web 服务器不是 Apache，那么请放在等效的文件中）：

```
AddType video/ogg .ogg
AddType video/mp4 .mp4
AddType video/webm .webm
```

如果这种方法不能够解决问题，您可以联系主机或服务器的管理员，查看服务器使用的 MIME 类型是否正确。如果您想了解如何设置 Web 服务器的其他类型，请从 Mozilla Developer Network¹中阅读文章“Properly Configuring Server MIME Types”。

¹ https://developer.mozilla.org/en/properly_configuring_server_mime_types



什么是.htaccess 文件?

在使用 Apache 的 Web 服务器时, .htaccess 文件提供了一种在目录基础上更改配置的一种方式。在 .htaccess 文件中的指令应用于所在的目录中以及所有的子目录中。关于 .htaccess 文件的更多信息, 请参见 [Apache documentation](#)¹。

5.4 用于网络的视频文件解码

我们为 The HTML5 Herald 所写的代码几乎是完美的, 它几乎能够使浏览网页的每个人都能够观看视频。因为我们需要将视频解码为两种格式(如果您愿意, 也可以是 3 个), 所以我们需要一个简单的方法将原视频文件编码为 HTML5 的预设格式。幸运的是, 有一些在线资源和桌面应用程序使您能做到这一点。

Miro Video Converter²是一个免费软件, 并且界面超级简单, 它能够使您将视频编码成为所需要的 HTML5 视频格式。它可用于 Mac 和 Windows³。

只需将文件拖到窗口, 或用习惯的方式浏览一个文件。一个下拉框可以提供选择, 您可以将视频编码为 Theora、WebM 或 MPEG-4 格式, 还有一个 MP3 的选项以及许多设备特定的视频输出的预先装置。

还有许多其他编码 HTML5 视频的选项, 这应该足以帮助您创建两个(或三个)所需文件用于嵌套视频, 使 99% 以上的用户可以观看。

5.5 创建自定义控件

相比用第三方技术嵌套视频的习惯方法, 使用 HTML5 视频有另一个巨大的好处。使用 HTML5 视频, video 元素便成为了网页真正的一部分, 而不是不可访问的插件。在很大程度上它是网页的一部分, 就像 img 元素或任何其他本机的 HTML 元素。这意味着我们可以使用 JavaScript 规定 video 元素和它的各种组件——我们甚至可以用 CSS 设计 video 元素。

¹ <http://httpd.apache.org/docs/1.3/howto/htaccess.html>

² <http://www.mirovideoconverter.com/>

³ Linux 用户可以使用 FFmpeg [<http://ffmpeg.org/>], Miro Video Converter 基于的命令行实用工具

如前所述，支持 HTML5 视频的每一个浏览器都会嵌套一组本机控件来帮助用户访问视频内容。在每一个浏览器中，这些控件都有不同的外观，它可能会使那些关注网站品牌的人感到困惑。没有问题：通过使用由视频元素提供的 JavaScript API，我们可以创建自定义的控件，并将它们链接到视频的运行中。

用任何元素都可创建自定义控件——图像，普通的 HTML 和 CSS，甚至是用 Canvas API 所画的元素——随您选择。要利用这种 API 创建自己的自定义控件，并将它们插入到页面中，然后使用 JavaScript 将那些静态图像元素转换为动态的、完全发挥性能的视频控件。

5.5.1 让我们从一些标记和设计开始

对于我们的示例网站，我们将创建一组非常简单的视频控件来展示新的 HTML5 视频 API 的强大功能。图 5.5 是一张截图，它显示了一组用于操作视频的控件。



图 5.5 我们将要创建的一组简单的视频控件

这两个按钮都有另外的状态：图 5.6 显示了视频在播放以及静音时所呈现的状态。

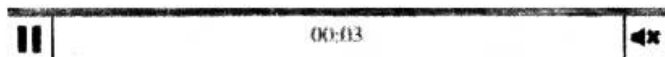


图 5.6 上面的控件在静音和播放时的状态

此控件有以下 3 个组件。

- 播放/暂停按钮。
- 从零计数的计时器。
- 静音/取消静音按钮。

在多数情况下，自定义控件应有各种浏览器本机提供的默认控件的所有功能。如果您的控件集引入了较少的或劣质的功能，将极有可能最终使用户感到沮丧。

对于引入 API 的目的，不是试图模仿浏览器本机提供的功能，我们要逐步介绍视频 API 的重要部分。这将在建立我们所工作的平台时使您能够亲身体验。

我们将为视频创建一个非常简单但十分实用的一组控件。我们的控件所缺少的主要功能是搜索栏，它可以使用户在视频处“滑动”，以找到视频的特定部分。这意味着除了刷新页面，没有其他办法返回视频的开头。除此之外，控件功能齐全——可

使用户播放、暂停视频，以及静音、取消静音。

以下是我们将使用的 HTML 代码来呈现视频控件的不同部分：

index.html (excerpt)

```
<div id="controls" class="hidden">
  <a id="playPause">Play/Pause</a>
  <span id="timer">00:00</span>
  <a id="muteUnmute">Mute/Unmute</a>
</div>
```

我们将不在此详细介绍 CSS，但有一份我们所做的总结（如果您想了解它是如何组成的，可以在文件夹中查看演示页面的源代码）：

- ❶ 可使用 `text-indent` 属性从查看中删除播放/暂停以及静音/取消静音按钮的文字。
- ❷ 一个单独的 CSS 精灵图像作为背景图像，可以表示按钮的不同状态（播放、暂停、静音、取消静音）。
- ❸ CSS 类用于表示不同的状态，可以使用 JavaScript 添加或删除这些类。
- ❹ 控件包装元素以绝对位置的方式放置，以覆盖视频的底部。
- ❺ 我们设定控件的默认透明度级别为 50%，但在鼠标悬停时，不透明度增加至 100%（我们将在第 6 章详细介绍不透明度）。
- ❻ 默认情况下，控件包装元素设为 `display: none`，使用 `hidden class` 设置为 `none`，我们将用 JavaScript 删除它。

如果您一直跟着所建立的示例网站，那么接下来设计 3 个您喜欢的元素。在这里我们将要完成的任务中，控件的外观是次要的，所以您可以随意摆弄，直到满意为止。

5.5.2 介绍媒体元素 API

让我们讨论一下创建自定义控件所需要的步骤，并且在此过程中，我们将向您介绍视频 API 的方方面面。随后，我们将从不在我们控件中使用的 API 中总结一些其他方法和功能，所以这是一个很好的概述，您可以看到 API 所包含的内容。

为了能够使这些新的自定义控件运行，我们首先会把它们放在 JavaScript 的变

量中，将其存放在高速缓存中。以下是代码的前几行：

js/videoControls.js (excerpt)

```
var videoEl = $('video')[0],
    playPauseBtn = $('#playPause'),
    vidControls = $('#controls'),
    muteBtn = $('#muteUnmute'),
    timeHolder = $('#timer');
```

当然，没有必要将我们的选择都放到高速缓存中，但是它通常是与高速缓存对象共同工作的最好做法（对于可维护性和性能），而不是针对页面上的各种元素，不必要地重复相同的代码。

第一行是针对 video 元素本身。当我们使用 API 时，会使用很多 videoEl 变量——由于大多 API 方法需要从媒体元素中调用。如果您注意过构成控件的 HTML 代码，那么下面的 4 行代码您将会非常熟悉。这是页面上的 4 个元素，我们将基于与用户的交互对其进行操作。

我们的首要任务是确保本机控件是隐藏的。我们可以轻松地做到这一点，只需简单地从 HTML 中删除控件属性即可。但由于我们的自定义控件依赖于 JavaScript，禁用 JavaScript 的用户将不能使用任何方式控制视频，因此我们将在 JavaScript 中删除控件属性。代码如下：

js/videoControls.js (excerpt)

```
videoEl.removeAttribute("controls");
```

下一步是使我们自定义的控件可见。如前所述，我们已经使用 CSS 的默认方式从查看中删除了控件。通过使用 JavaScript 使自定义控件可见，我们确保用户将不会看到两组控件。

所以下一段代码如下：

js/videoControls.js (excerpt)

```
videoEl.addEventListener('canplaythrough', function () {
    vidControls.removeClass("hidden");
}, false);
```

这是我们第一次使用 HTML5 视频 API 的功能。首先，请注意 addEventListener 方法。此方法恰如其名：它将侦听发生在目标元素的指定事件。



addEventListener 并不支持多种浏览器！

如果您熟悉支持多种浏览器的 Javascript 技术，您可能了解 `addEventListener` 方法不支持多种浏览器。在这种情况下，它不会构成问题。在我们使用的浏览器中缺乏对 `addEventListener` 支持的只有 Internet Explorer 9 之前的版本——这些浏览器根本就不支持 HTML5 视频。

我们所要做的是使用 Modernizr（或等效的 JavaScript）来检测对 HTML5 视频 API 的支持，然后仅为提供支持的浏览器运行代码——它们都将支持 `addEventListener`。

在这种情况下，我们针对的是 `video` 元素的本身。我们注册侦听的是视频 API 的 `canplaythrough` 事件。根据在规范中此事件的定义¹：

用户代理判断，如果现在开始播放，将以目前的播放率呈现媒体资源直到结束，并且不会为进一步的缓冲而停止。

我们还可以使用其他事件检测视频是否已经准备就绪，每一个事件都有它的特殊目的。我们稍后将在本章接触一些其他的事件。这个特殊的事件将保证视频连续播放，所以如果我们希望视频流畅稳定地播放，使用它是非常适合的。

5.5.3 播放和暂停视频

当引发 `canplaythrough` 事件时，运行回调函数。在该函数中，我们已经写入了单独一行代码，用以从控件包装器中删除 `hidden` 类，所以我们的控件是可见的。现在我们想为控件添加一些功能。让我们将 `click` 事件处理程序与我们的播放/暂停按钮捆绑起来：

js/videoControls.js (excerpt)

```
playPauseBtn.bind('click', function () {
  if (videoEl.paused) {
    videoEl.play();
  } else {
    videoEl.pause();
  }
});
```

¹ <http://www.whatwg.org/specs/web-apps/current-work/multipage/video.html#event-media-canplaythrough>

当单击按钮时，我们运行 `if/else` 代码块，将会使用视频 API 的 3 个其他功能。以下是对这 3 个功能的描述。

(1) 访问 `paused` 属性可以看到当前的视频是否处于“暂停”状态。这并不是意味着用户暂停了该视频，它可以同样仅表示视频播放之前处在起始的状态。所以如果视频当前没有播放，该属性将返回 `true`。

(2) 由于我们现在已经决定单击播放/暂停按钮，并且视频当前没有处于播放状态，我们可以安全地调用 `video` 元素的 `play()` 方法。这将从上次暂停的位置播放视频。

(3) 最后，如果 `paused` 属性没有返回 `true`，将会引发 `else` 的代码部分，这样将会启动 `video` 元素的 `pause()` 方法，停止视频。

您可能已经注意到我们的自定义控件没有“停止”按钮（一般习惯用正方形图标表示）。如果您觉得有必要，可以添加这样一个按钮，但是由于搜索栏可以移动到视频的开头，因此许多视频播放器不使用它。唯一美中不足的是视频 API 没有“停止”方法。为了解决这个问题，您可以通过暂停，使视频模仿“停止”状态，然后将它发送到开始的位置（稍后会作详细介绍）。

您可能已经注意到在我们的 `if/else` 结构中缺失了一些东西。早先，我们已经向您展示了几张截图，截图显示了控件的两种状态。我们需要使用 JavaScript 改变精灵图像的背景位置，我将按钮“play me”改为“pause me”。

代码如下：

`js/videoControls.js (excerpt)`

```
videoEl.addEventListener('play', function () {
    playPauseBtn.addClass("playing");
}, false);
videoEl.addEventListener('pause', function () {
    playPauseBtn.removeClass("playing");
}, false);
```

在这里，`addEventListener` 方法有两种以上的应用（如果您使用视频和音频的 APIs，就必须熟悉它）。第一段代码块用于侦听 `play` 事件。所以，如果我们所写的单击处理程序触发了 `play()` 方法（或者，如果其他东西引起视频播放，例如如此页上的其他一些代码），那么监听器就会发现 `play` 事件并运行回调函数。

第二段代码块工作原理与第一段代码块相同，只是它侦听的是 `pause` 事件（不要与 `paused` 属性混淆）。

如果已经播放了该元素，第一段代码块将给播放/暂停按钮添加 class playing。该 class 将改变播放/暂停按钮上的精灵图像的背景位置，以使“pause me”图标出现。同样，第二段代码块将删除 playing class，使按钮的状态恢复到默认状态（“play me”状态）。

您也许会想，“为什么不只在代码中添加或删除 playing class 来处理单击按钮？”当单击按钮或通过键盘访问时，都没有问题，但是在这里我们需要考虑另外一种运行状态，如图 5.7 所示。

当设置 video 元素的快捷菜单后，上面的菜单就会出现。正如您所看到的，单击 video 元素的控件并不是播放/暂停或静音/取消静音的唯一途径。



图 5.7 可以通过快捷菜单访问一些视频控件

为了能够以任何方式访问 video 元素的功能时都能够确保改变按钮的状态，我们将不再侦听 play 和 pause 事件（稍后，您将看到 sound-related 事件），用以改变按钮的状态。



禁用快捷菜单

您可能也关注到了，video 元素的快捷菜单有一个 Save Video As 的选项。网络上也有一些关于如此简便保存 HTML5 视频的讨论，并且这也影响到了如何传播受版权保护的视频。一些内容制作人可能只是由于这个原因而避免使用 HTML5 视频。

无论您怎样做，都要认识到与网络视频相关的实际存在的事物。多数试图复制或传播受版权保护的视频的用户，无论对视频采取何种保护措施，他们都会想尽办法达到目的。有一些 Web 应用程序和软件工具可以轻易地剥离甚至是基于 Flash 的视频。您也应该知道，即使您禁用了 video 元素的快捷菜单，用户仍可以看到页面的源代码，并找到视频文件的位置。

一些网站，如 YouTube，在使用 HTML5 视频时，已经应用了一些程序打击此类行为。YouTube 的页面允许您选择进入到他们的 HTML5 视频试用版¹。选择进入后，在您观看视频以及打开 video 元素的快捷菜单时，将有一个自定义快捷菜单。仍然显示“Save Video As...”选项。但速度不是很快！如果您选择此选项，（到编写本书时），那么您就被他们“晃点”²了。很狡诈！

¹ <http://www.youtube.com/html5>

² <http://en.wikipedia.org/wiki/Rickrolling>

YouTube 也在页面上动态添加 video 元素，这样您就不能在源文件中找到视频文件的 URL。

所以要意识到，您可以有自己的选择，并且可以使剥离受版权保护的视频变得更加困难。但也必须认识到改变用户的初衷会有一些弊端，为了一些可能很小的问题，而要编写复杂的脚本及标记，以及与之相关的性能及维护问题，即使有这些问题，但还是能够从中受益。

5.5.4 视频音轨的静音与取消静音

在我们的脚本中所要添加的另一项功能是静音/取消静音按钮。这段代码与用于播放/暂停按钮的代码几乎相同。现在，我们已经将 click 事件捆绑到静音/取消静音的按钮上，接下来是一个非常相似的 if/else 结构：

js/videoControls.js (excerpt)

```
muteBtn.bind('click', function () {
  if (videoEl.muted) {
    videoEl.muted = false;
  } else {
    videoEl.muted = true;
  }
});
```

此段代码引入了 API 的一个新内容：muted 属性。单击静音按钮以后，我们来检测一下此属性的状态。如果是 true（意味着是静音），我们将它设置为 false（取消静音）；如果是 false，我们将它的状态设置为 true。

此外，我们在这里没有处理按钮的状态，原因与早先讨论过的播放/暂停按钮一样：快捷菜单允许设置静音和取消静音，所以，我们根据视频的实际静音或取消静音的状况来改变按钮的状态，而不是单击按钮。

但是不像播放/暂停按钮，我们没有能力去侦听 mute 和 unmute 事件。反而，API 提供了 volumechange 事件：

js/videoControls.js (excerpt)

```
videoEl.addEventListener('volumechange', function () {
  if (videoEl.muted) {
    muteBtn.addClass("muted");
  } else {
    muteBtn.removeClass("muted");
  }
}, false);
```


此外，每次指定事件发生（在此例中是改变音量）时，我们使用事件侦听器运行一些代码。您可能已经推断出了此事件的名字，`volumechange` 事件不仅限于检测静音和取消静音，它也可以检测到音量的变化。

一旦检测到音量的变化，我们将检测 `video` 元素的 `muted` 属性，并且相应地改变静音/取消静音按钮的 `class`。

5.5.5 视频结束播放的响应

到目前为止，我们所写的代码允许用户播放和暂停视频，也可以静音和取消静音。这些都使用了我们的自定义控件。

这时，如果使视频播放到结尾处，它将会停在最后一帧。我们认为最好是将视频发送回第一帧，准备再次播放。这样我们有机会引入两个 API 的新功能：

`js/videoControls.js (excerpt)`

```
videoEl.addEventListener('ended', function () {
    videoEl.currentTime = 0;
    videoEl.pause();
}, false);
```

此代码块侦听 `ended` 事件，它将告诉视频已到结尾处并已经停止。一旦检测到此事件，我们就将视频的 `currentTime` 属性设为零。此属性表示当前的播放位置，以秒（十进制小数）为单位。

它将我们的代码引领到下一步。

5.5.6 更新视频播放的时间

现在到了最后一步：我们想让计时器在视频播放时更新当前的播放时间。我们已经引入了 `currentTime` 属性，可以使用它更新 `#timeHolder` 元素的内容。代码如下：

`js/videoControls.js (excerpt)`

```
videoEl.addEventListener('timeupdate', function () {
    timeHolder[0].innerHTML = secondsToTime(videoEl.currentTime);
}, false);
```

在这种情况下，我们侦听的是 `timeupdate` 事件。`timeupdate` 事件每一次

都会引发视频时间的改变，这就意味着即使是转瞬间的变化，也会引发此事件。

这已经足够创建一个基本的计时器。不幸的是，显示的时间是不予与合作的，由于您所看到的时间每毫秒都在变化，会产生无数的小数位，所以看上去很难看，如图 5.8 所示。

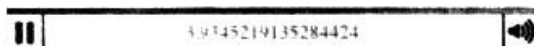


图 5.8 在 HTML 中直接使用 `currentTime` 属性不是很理想

另外，在这种状态下的计时器将不显示分钟和小时，仅显示秒——根据视频的长度，结束时可能会是成百上千。这是不切实际的，至少可以这样说。

将秒的格式转换为一个更人性化的时间表现形式，我们已经编写了一段函数，叫做 `secondsToTime()`，并且从上面的 `timeupdate` 处理程序调用它。在本例中，我们不想显示毫秒，所以我们的函数将计时器四舍五入到秒。以下是函数的开始部分：

js/videoControls.js (excerpt)

```
var h = Math.floor(s / (60 * 60)),
    dm = s % (60 * 60),
    m = Math.floor(dm / 60),
    ds = dm % 60,
    secs = Math.ceil(ds);
```

写入这 5 行代码后，最终变量 `secs` 将会存有一个四舍五入以秒计算的数字，它是由传到函数中的秒数计算而得的。

下一步，我们需要确保一个以秒或分钟为单位的单独个位数字用 05 表示，而不仅仅是 5。下一段代码块将解决此问题：

js/videoControls.js (excerpt)

```
if (secs === 60) {
    secs = 0;
    m = m + 1;
}

if (secs < 10) {
    secs = "0" + secs;
}
```



```

if (m === 60) {
    m = 0;
    h = h + 1;
}

if (m < 10) {
    m = "0" + m;
}

```

最后，我们返回一个字符串以正确的格式表示当前的视频时间：

js/videoControls.js (excerpt)

```

if (h === 0) {
    fulltime = m + ":" + secs;
} else {
    fulltime = h + ":" + m + ":" + secs;
}

return fulltime;

```

包含 if/else 结构是为了检测视频播放的时间是否是一个小时或者更长。如果是这样，我们将使用两个冒号的时间格式。否则，将使用一个冒号的时间格式将分钟和秒分开，大多数情况是这样。

请记住，我们是在哪里运行的这个函数。我们将它包含在 timeupdate 事件处理程序中。函数所返回的结果将成为 timeHolder 元素的内容（它是一个有 timer 的 id 的缓存元素）：

js/videoControls.js (excerpt)

```
timeHolder[0].innerHTML = secondsToTime(videoEl.currentTime);
```

由于 timeupdate 事件是由每个瞬间变换引发的，因此 timeholder 元素的内容变换非常快。但是由于我们将值四舍五入近似到秒，因此可见的变换仅限于每秒的时间更新，即使在技术上时间元素内容的变化更快一些。

就是这样，我们完成了自定义控件。按钮能够按照预期工作，定时器运行顺利。正如我们上面提到的，这不是功能齐全的一组控件。但是您至少现在能够从 JavaScript 过渡到在 HTML5 视频交互的基础上对其进行一些很好的控制，所以，现在可以进行一些修补，并看看还有什么需要添加的。

5.5.7 媒体元素 API 的其他一些功能

API 所包含的内容远远多于我们在此介绍的内容。这里是关于在创建自己的自定义控件或使用音视频元素时可能用到的一些事件和属性的总结。

有一点需要记住,即 API 方法和属性可以在 JavaScript 中的任何地方使用——它们不需要链接到自定义控件。如果您喜欢在鼠标悬在上面时播放视频,或使用 audio 元素播放各种与 Web 应用程序或游戏相关的声音,您所要做的就是调用适当的方法。

事件

我们已经学习了 canplaythrough, play, pause, volumechange, ended 和 timeupdate 事件。还有下面的一些事件可以在处理 HTML5 视频和音频时使用。

Canplay

它与 canplaythrough 相似,但是,只要视频可以播放,即使是几帧,也会引发它。(它与 canplaythrough 相反,您应记住,只有浏览器认为可以以同样的方式播放视频到结尾处而不需重新缓冲时才引发此事件。)

error

当错误发生时,才发送这个事件。另外,还有一个 error 属性。

loadeddata

加载媒体的第一帧。

loadedmetadata

当加载完毕媒体元数据时,发送此事件。它将包括尺寸、持续时间和文本轨道(副标题)。

playing

它表明媒体刚开始播放。playing 与 play 的区别在于:视频循环或再一次播放开始时,将不会发送 play 事件,但会发送 playing 事件。

seeking

当搜索操作开始时,将发送此事件。当用户开始移动搜索栏选择新的视频或音

频部分时，可能会引发该事件。

seeked

当搜索操作完成时，引发该事件。

功能

除了我们学习过的功能，这里还有其他一些对您非常有用处的功能。

playbackRate

默认播放率是 1。对它进行更改而加速或减缓播放。如果您正在创建一个快进或倒带按钮，或慢动作/慢速倒带按钮，自然是非常实用的。

src

正如名字的暗示，这个属性返回的是指向播放视频的 URL。它只有在 video 元素上使用 src 属性时才工作。

currentSrc

它将返回指向播放视频文件的 URL 的值，不管它是来自于 video 元素的 src 属性或 source 元素之一。

readyState

此属性返回一个从 0 到 4 的数字值，每一个状态表示一个媒体元素的准备等级。例如，“1”表示媒体的元数据已经可用。“4”几乎与引发 canplaythrough 事件的条件相同，意味着视频已经准备好播放，并且不会因为缓冲或加载打断。

duration

此事件返回视频短时间内的长度。

buffered

它表示视频已经缓冲并可供浏览器播放的时间范围。

videoWidth, videoHeight

这些值返回视频的原有尺寸，编码视频时的实际宽度和高度——不是声明在 HTML 和 CSS 中的。可以通过常用的 width 和 height 属性访问这些值。

您也可以访问那些直接声明在 HTML 中的属性，例如，preload, controls,

autoplay, loop 和 poster。

5.6 关于音频

我们所讨论的关于 HTML5 视频和 API 的许多内容同样应用于 audio 元素，而明显的区分是它们与视觉相关。

与 video 元素类似，preload, autoplay, loop, 和 controls 属性可用于（或不可用于）audio 元素。

视频元素不显示任何东西，除非显示控件，即使元素控件不存在，仍可通过脚本访问元素。如果您想在网站中使用声音，但不想将它捆绑在控件上展示给用户，那么它将是非常有用的。audio 元素嵌套了 source 的标签，与视频相似，它也将任何不是 source 标签的子代元素当作不提供支持的浏览器的后备内容。

关于对编解码器/格式的支持，Firefox, Opera 和 Chrome 都支持 Ogg/Vorbis; Safari, Chrome 和 Internet Explorer 9 都支持 MP3; 每一个提供支持的浏览器都支持 WAV。Safari 也支持 AIFF。目前，MP3 和 Ogg/Vorbis 将足以应付所有提供支持的浏览器。

5.7 可访问的媒体

除了其作为页面一等公民的地位，还要能够更多地从键盘访问它们（例如，使用 tabindex），HTML5 元素也能够使您访问 track 元素，以显示正在播放的视频文件的副标题或文本。就像 source 元素，应将 track 元素作为 video 或 audio 元素的子代。

track 元素仍在不断发展中，但是如果作为 video 元素的子代包含它，那么它应像下面的示例所显示的一样（与规范中所给出的示例相近）：

```
<video src="brave.webm">
  <track kind="subtitles" src="brave.en.vtt" srclang="en"
  ➤label="English">
  <track kind="captions" src="brave.en.vtt" srclang="en"
  ➤label="English for the Hard of Hearing">
  <track kind="subtitles" src="brave.fr.vtt" srclang="fr"
  ➤label="Français">
  <track kind="subtitles" src="brave.de.vtt" srclang="de"
  ➤label="Deutsch">
</video>
```

此代码包含 4 个 `track` 元素, 每个元素为不同语言的副标题引用一个文本轨迹 (或者, 在此例中的第二个 `track` 元素中, 可以是同一种语言的替代内容)。

`kind` 属性可以取值为 5 个值中的任何一个值: `subtitles`、`captions`、`descriptions`、`chapter` 和 `metadata`。`src` 属性是必需的, 并且它指向包含轨迹信息的外部文件。`srclang` 属性规定了语言。最后, `label` 属性为轨迹给出了用户可读的标题。

编写本书时, `track` 元素还没有得到任何浏览器的支持。关于此新元素的更多信息, 请参见 W3C 规范¹。

5.8 小结

网络上的视频和音频早已被 Flash 占据, 但是, 正如我们所看到的, HTML5 正在改变它。虽然目前的编解码器和格式景观还不太完整, 但是, 完全脚本化的多媒体内容的承诺, 以及在浏览器本机运行视频音频而取代了插件包装器的性能优势, 都极大地吸引了网络设计师、开发人员和内容提供商。

由于我们使用了十分安全的备用技术, 因此现在没有任何理由不开始尝试使用这些新元素。至少, 当更普遍地提供对此技术的支持时, 我们应该做好相应的准备。

我们仅介绍了关于 HTML5 的“特有”部分 (在规范中, 这只是一小部分)。在接下来的几章中, 我们将我们的注意力转向 CSS3, 并开始使我们的 *The HTML5 Herald* 看起来十分精美。最后, 我们将看一下经常与“HTML5”术语捆绑在一起的新 JavaScript API。

¹ <http://dev.w3.org/html5/spec/Overview.html#the-track-element>

CSS3 简介

内容层已经完成，现在是时候对它进行一下美化了。接下来的 4 章将主要介绍显示。本章介绍一些基本知识：我们首先快速简单地介绍 CSS 选择器，了解 CSS3 中增加了哪些新功能。然后介绍一些指定颜色的新方式。接下来详细介绍圆角、投影和文本阴影，以及一些使我们无需创建数十个圆角和文本图像来匹配设计即可设置页面样式的提示和技巧。

但是首先，需要确保旧浏览器能够识别页面上的新元素，以便我们能够设置它们的样式。

6.1 改进旧版浏览器

第 1 章中曾提到，要在旧版的 Internet Explorer 中设置新 HTML5 元素的样式，需要一个称为 HTML5 Shiv 的 JavaScript 片段。也可以使用附录 A 中详细介绍的 Modernizr 库（其中包含一段类似的代码）。

但是，即使有了这段 JavaScript，也不一定能够开始。现在 Internet Explorer 6 到 Internet Explorer 8 能够感知这些新元素，但它们仍然缺乏一些默认样式。事实上，其他浏览器的早期版本也会遇到这种情况。虽然它们可能允许任意元素，但是它们无法知道 `article` 应该是块级别的而 `mark` 应该是内联的。因为元素默认以内联方式呈现，所以最好告诉这些浏览器哪些元素应该是块级别的。

这可以使用简单的 CSS 规则来完成:

css/styles.css (excerpt)

```
article, aside, figure, footer, header, hgroup, nav, section {
    display: block;
}
```

有了此 CSS 和需要的 JavaScript, 所有浏览器将能够平等地设置 HTML5 元素的样式。

6.2 CSS3 选择器

选择器是 CSS 的核心。没有选择器来定位页面上的元素, 就只能使用元素的 style 属性并内联地声明样式来修改元素的 CSS 属性。当然, 这么做单调、麻烦还难以维护, 所以我们使用选择器。起初, CSS 允许按类型、类和/或 id 匹配元素。这需要向标记添加 class 和 id 属性来创建挂钩, 并区分同一类型的不同元素。CSS2.1 添加了伪元素、伪类和选择符。使用 CSS3, 可以使用丰富的选择器来定位页面上的所有元素。

下面的介绍中将包含以前的 CSS 版本为我们提供的选择器。包含它们是因为, 虽然现在可以开始使用 CSS3 选择器, 但是仍支持以前的 CSS 版本的所有选择器。即使是已存在很长时间的选器, 在这里也值得回顾一下, 因为对其中许多选择器的浏览器支持才刚刚达到可用的程度。

6.2.1 关系选择器

关系选择器基于标记中元素的彼此关系来定位它们。Internet Explorer 7+、Firefox、Opera 和 WebKit 都支持这些选择器。

后代选择器 (E F)

您一定很熟悉这个选择器。后代选择器定位作为元素 E 的后代 (子、孙子、曾孙等) 的任何元素 F。例如, ol li 定位有序列表中的 li 元素。这将包括嵌套在一个 ol 中的 ul 中的 li 元素——您可能不希望这么做。

子选择器 (E > F)

此选择器匹配作为元素 E 的直接子元素的任何元素 F——将忽略任何进一步嵌

套的元素。继续看上面的示例，`ol > li` 将仅定位 `ol` 中直接包含的 `li` 元素，忽略嵌套在 `ul` 中的元素。

相邻兄弟选择器 (E + F)

此选择器将匹配与 `E` 具有相同父元素并在标记中紧挨着 `E` 之后的任何元素 `F`。例如，`li + li` 将定位指定容器中除第一个 `li` 之外的所有 `li` 元素。

一般兄弟选择器 (E ~ F)

此选择器稍微复杂一点。它将匹配与任何 `E` 元素具有相同父元素且在标记中位于它之后的任何元素 `F`。所以，`h1~h2` 将匹配 `h1` 之后的任何 `h2`，只要它们具有相同的直接父元素——也就是说，只要 `h2` 没有嵌套在任何其他元素中。让我们看一个简单示例：

```
<article>
  <header>
    <h1>Main title</h1>
    <h2>This subtitle is matched </h2>
  </header>
  <p> blah, blah, blah ...</p>
  <h2>This is not matched by h1~h2, but is by header~h2</h2>
  <p> blah, blah, blah ...</p>
</article>
```

选择器字符串 `h1~h2` 将匹配第一个 `h2`，因为它们都是 `header` 的子元素或直接后代。第二个 `h2` 不匹配，因为它的父元素是 `article`，而不是 `header`。但是，它将与 `header~h2` 匹配。类似地，`h2~p` 仅匹配最后一段，因为第一段在与它具有共同父元素 `article` 的 `h2` 之前。



没有反向选择器

您将会注意到没有“父”或“祖先”选择器，也没有“前辈兄弟”选择器。这有时可能不太令人满意，但这么做是有原因的：在决定是否应用一种样式之前，如果浏览器必须在 DOM 树中向上回溯，或者递归到一组嵌套的元素中，则呈现将明显变慢并需要更多的处理能力。请访问 http://snook.ca/archives/html_and_css/css-parent-selectors，了解此问题的更详细说明。

仔细查看 The HTML5 Herald 的样式表，将会看到我们在许多位置使用了这些选择器。例如，当确定网站的总体布局时，我们希望 3 栏 `div` 浮动在左侧。为了避

- 免将此样式应用到它们内部嵌套的任何其他 div，我们使用子选择器：

css/styles.css (excerpt)

```
#main > div {
    float: left;
    overflow: hidden;
}
```

在后面几章中向网站添加新样式时，将会看到其中许多选择器类型。

6.2.2 属性选择器

CSS2 引入了一些属性选择器。这些选择器可基于元素的属性来匹配元素。CSS3 扩展了这些属性选择器，支持基于模式匹配来定位元素。

E[attr]

将具有属性 attr 的任何元素 E 与任意值相匹配。第 4 章中曾使用此选择器来设置所需输入的样式——input:required 适用于最新的浏览器，但 input[required] 具有相同的效果并适用于一些较老的浏览器。

E[attr=val]

将具有属性 attr 的任何元素 E 与区分大小写的值 val 准确匹配。尽管不是新选择器，但它在定位表单输入类型（比如，使用 input[type=checkbox] 定位复选框）时很有帮助。

E[attr|=val]

匹配其属性 attr 具有值 val 或以 val- 开始的任何元素 E。此选择器最常用于 lang 属性（比如 lang="en-us"）。例如，p[lang|= "en"] 将匹配定义为英语的任何段落，无论是英式英语还是美式英语。

E[attr~=val]

匹配其属性 attr 的值中包含两边有空格的完整单词 val 的任何元素 E。例如，.info[title~="more"] 将匹配其 class info 包含一个 title 属性，而该属性又包含单词“more”的任何元素，比如“Click here for more information”。

E[attr^=val]（Internet Explorer 8 及更高版本、WebKit、Opera、Mozilla）

匹配其属性 attr 以值 val 开头的任何元素 E。换句话说，val 与属性值的开

头相匹配。

E[attr\$=val] (Internet Explorer 8 及更高版本、WebKit、Opera、Mozilla)

匹配其属性 attr 以 val 结束的任何属性 E。换句话说, val 与属性值的末尾相匹配。

E[attr*=val] (Internet Explorer 8 及更高版本、WebKit、Opera、Mozilla)

匹配其属性 attr 内任意位置包含 val 的任何元素 E。换句话说, 字符串 val 与属性值中的任意位置相匹配。它类似于上面的 E[attr~=val], 但 val 可以是一个词的一部分。使用与上面相同的示例, .fakelink[title~=info]{} 将匹配其 class fakelink 具有 title 属性并且该属性又包含字符串 info 的任何元素, 比如 “Click here for more information”。

6.2.3 伪类

可能您已经熟悉一些用户交互伪类, 比如:link、:visited、:hover、:active 和: focus。



要记住的要点

(1) :visited 伪类可能带来安全风险, 可能在未来不会得到全面支持。简单地讲, 恶意网站可向 visited 链接应用一种样式, 然后使用 JavaScript 检查流行网站的链接样式。这使攻击者无需用户的权限即可窥视他们的浏览历史。因此, 一些浏览器已开始限制可使用:visited 应用的样式, 其他一些浏览器 (Safari 5 最明显) 已完全禁用它。

规范明确允许这些更改: “UA[用户代理]可以因此将所有链接视为未访问的链接, 或者实现其他措施来保护用户的隐私, 同时以不同方式呈现已访问和未访问的链接。”

(2) 为了实现更高的可访问性, 可以在包含: hover 的地方添加: focus, 因为不是所有访问者都将使用鼠标来导航网站。

(3) :hover 可应用到页面的任何元素, 而不仅仅是链接和表单控件。

(4) :focus 和: active 与链接、表单控件和任何具有 tabindex 属性的元素相关。

虽然您可能已使用这些基本伪类有一段时间了, 但是还有其他许多伪类可用。

其中一些多年前就已备案到规范中，但直到浏览器开始支持新 HTML5 表单属性（这些属性可使它们更有意义）后才得以支持（或为公众所知）。

以下伪类基于属性、用户交互和表单控件状态来匹配元素。

:enabled

一个已启用的用户界面元素。

:disabled

一个已禁用的用户界面元素。

:checked

已选择或勾选的单选按钮或复选框。

:indeterminate

既没有选中也没有取消的表单元素。这个伪类仍存在争议，可能未来会包含在规范中。

:target

此选择器挑选作为当前活动页面内定位点目标的元素。这实际上没有听起来那么复杂：您已经知道可以在目标的 id 中使用 # 字符，获取页面中的定位点的链接。例如，页面中可以有这样一个链接 `Skip to content`，当单击时，它将跳到 id 为 content 的元素。

这会将地址栏中的 URL 更改为 `thispage.html#content`，并且 `:target` 选择器现在匹配 `#content` 元素，就好像已临时包含了选择器 `#content`。我们说“临时”是因为，只要用户单击不同的定位点，`:target` 就会匹配新的目标。

:default

应用于一个或多个作为一组类似元素中的默认元素的 UI 元素。

:valid

应用于有效的元素（基于 `type` 或 `pattern` 属性，如第 4 章所述）。

:invalid

应用于空的必填元素，以及未能与 `type` 或 `pattern` 属性所定义的需求相匹

配的元素。

:in-range

应用于具有范围限制的元素，其中该值位于这些限制内。例如，这适用于具有 `min` 和 `max` 属性的 `number` 和 `range` 输入类型。

:out-of-range

与 `:in-range` 元素相反，它们的值在限制范围外。

:required

应用于具有 `required` 属性集的表单控件。

:optional

应用于没有 `required` 属性的所有表单控件。

:read-only

应用于其内容无法供用户修改的元素。这包括除表单字段以外的大部分元素。

:read-write

应用于其内容可供用户修改的元素，比如文本输入字段。

浏览器对这些伪类的支持不尽相同，但这类支持正在迅速完善。支持表单控件属性（比如 `required` 和 `pattern`）的浏览器也支持相关的 `:valid` 和 `:invalid` 伪类。

Internet Explorer 6 无法理解除链接以外的其他元素上的 `:hover`，Internet Explorer 6 和 Internet Explorer 7 都无法理解 `:focus`。Internet Explorer 8 和更早的版本缺乏对 `:checked`、`:enabled`、`:disabled` 和 `:target` 的支持。好消息是 Internet Explorer 9 支持这些选择器。

虽然仍然缺乏支持，但是 jQuery 等 JavaScript 库可帮助在不受支持的浏览器中定位这些伪类。

6.2.4 结构化伪类

目前为止，我们介绍了如何基于目标元素的属性和状态来定位它们。CSS3 还

支持简单地基于元素在标记中的位置来定位它们。这些选择器分组到结构化伪类¹类别下。

这些伪类目前可能看起来很复杂，但稍后在介绍应用它们的方式时，就会发现它们很容易理解。Internet Explorer 9 以及所有其他浏览器的最新和较老版本（但不包括 Internet Explorer 8 和更早版本）都支持这些选择器。

:root

根元素，它始终是 html 元素。

E F:nth-child(n)

作为父元素 E 的第 n 个子元素的元素 F。

E F:nth-last-child(n)

作为父元素 E 的倒数第 n 个子元素的元素 F。li:nth-last-child(1) 将匹配任何列表中的最后一项，与 li:last-child 相同（参见下文）。

E:nth-of-type(n)

指定父元素内具有指定类型的第 n 个元素。

E:nth-last-of-type(n)

类似于 nth-of-type(n)，但从父元素中最后一个元素向前计算。

E:first-child

作为父元素的第一个子元素的元素 E。这与:nth-child(1) 相同。

E:last-child

作为父元素中最后一个子元素的元素 E，与:nth-last-child(1) 相同。

E:first-of-type

与:nth-of-type(1) 相同。

E:last-of-type

与:nth-last-of-type(1) 相同。

¹ <http://www.w3.org/TR/css3-selectors/#structural-pseudos>

only-child

父元素中唯一的子元素。

E:only-of-type

父元素中唯一具有该类型的元素。

E:empty

没有子元素的元素，包括文本节点，所以<p>hello</p>不会匹配。

E:lang(en)

具有使用双字母缩写（en）表示的语言的元素。

E:not(exception)

这个伪类尤其有用，它将选择与括号内的选择器不匹配的元素。

包含:not 伪类的选择器匹配冒号左侧内容的所有元素，然后从匹配组中排除与冒号右侧内容匹配的元素。首先对左侧进行匹配。例如，`p:not(.copyright)`将首先匹配文档中的所有段落，然后从该集合中排除其class为copyright的所有段落。可以将几个:not 伪类连接起来形成一个字符串。`h2:not(header>h2):not(.logo)`将匹配页面上除了包含在header中且其class为logo的所有h2。



n 是什么？

有 4 个伪类接受括号中的 n 参数：`:nth-child(n)`、`:nth-last-child(n)`、`:nth-of-type(n)`和`:nth-last-of-type(n)`。

在最简单的情况下，n 可以是一个整数。例如，`:nth-of-type(1)`将定位一个系列中的第一个元素。也可以传递关键字 odd 或 even 中的一个，定位其他每个元素。还有一项更强大的功能，那就是传递一个数字表达式，比如`:nth-of-type(3n+1)`。3n 表示每隔 3 个元素，它定义了频率，而+1 是偏移。默认偏移为 0，所以`:nth-of-type(3n)`将匹配系列中第 3 个、第 6 个和第 9 个等元素，`:nth-of-type(3n+1)`将匹配第 1 个、第 4 个、第 7 个等元素。也允许使用负偏移值。

有了这些数值伪类，无需向标记中添加类，即可确定希望定位哪些元素。最常见的示例是一个表，其中每隔一行的颜色都较暗，以易于阅读。我们过去必须向每个 tr 添加 odd 或 even 类来实现此目的。现在，只需声明 `tr:nth-of-`

`type(odd)` 即可定位每个奇数行, 无需更改标记。甚至可以更进一步, 实现 3 色条带状表: 定位 `:nth-of-type(3n)`、`:nth-of-type(3n+1)` 和 `:nth-of-type(3n+2)` 并分别应用不同的颜色。

6.2.5 伪元素和生成的内容

除了伪类, CSS 还支持访问伪元素。伪元素可用于定位文档中包含的文本, 但无法在文档树中定位。伪类一般反映无法在 CSS 中轻松或可靠地检测到的某个元素属性或状态。另一方面, 伪元素表示 DOM 外部的某种文档结构。

例如, 所有文本节点都拥有第一个字母和第一行, 但如何定位它们而不将它们封装到 `span` 中呢? CSS 提供了 `::first-letter` 和 `::first-line` 伪元素, 它们分别匹配文本节点的第一个字母和第一行。它们仅使用一个冒号: `:first-line` 和 `:first-letter`。



为什么要麻烦地使用双冒号?

双冒号是正确的语法, 但单冒号受到了更好的支持。Internet Explorer 6、Internet Explorer 7 和 Internet Explorer 8 仅能理解单冒号表示法。所有其他浏览器都同时支持两种表示法。虽然 `:first-letter`、`:first-line`、`:first-child`、`:before` 和 `:after` 自 CSS2 就已经推出, 但是这些伪元素在 CSS3 中使用双冒号进行了重新定义, 以将它们与伪类相区分。

1. 生成的内容

`:before` 和 `:after` 伪元素并不是指存在于标记中的内容, 而是可以插入额外内容的位置, 这些内容在 CSS 中的该位置生成。尽管生成的该内容不会成为 DOM 的一部分, 但它可以设置样式。

要为伪元素生成内容, 可使用 `content` 属性。例如, 假设想要在页面上所有外部链接之后的括号中附上它们所指向的 URL, 以向用户明确表明他们将离开所在的页面。无需将 URL 硬编码到标记中, 可以结合使用一个属性选择器和 `:after` 伪元素:

```
a[href^=http]:after {
  content: " (" attr(href) ")";
}
```


`attr()` 可用于访问所选元素的任何属性，这对于显示链接目标很方便。回想一下属性选择器一节，`a[href^=http]` 表示“其 `href` 属性以 `http` 开头的任何 `a` 元素”，换句话说，就是外部链接。

下面是另一个示例：

```
a[href$=pdf] {
  background: transparent url(pdficon.gif) 0 50% no-repeat;
  padding-left: 20px;
}
a[href$=pdf]:after {
  content: " (PDF)";
}
```

这些样式将在 PDF 链接之后添加一个 PDF 图标和文本“(PDF)”。回想一下，`[attr$=val]` 选择器匹配属性的末尾，所以 `document.pdf` 将匹配，但 `page.html` 不会匹配。

2. `::selection`

`::selection` 伪元素匹配突出显示的文本。

WebKit 支持它，在 Firefox 中需要添加 `-moz` 供应商前缀。让我们在 The HTML5 Herald 上使用一下它，使选择背景和文本颜色与网站剩余部分的单色样式保持一致：

css/styles.css (excerpt)

```
::-moz-selection{
  background: #484848;
  color:#fff;
}
::selection {
  background:#484848;
  color:#fff;
}
```

6.3 CSS3 颜色

您一定急切地想将 CSS3 中真正实用的功能应用于实践，但是在这之前，我们还需要兜个圈子。CSS3 支持一些在页面上描述颜色的新方式。因为我们将后面几章的示例中使用它们，所以现在有必要介绍一下。

在 CSS3 之前，我们几乎总是使用十六进制格式（#FFF 或 #FFFFFF 表示白色）来声明颜色。也可以使用 `rgb()` 表示法声明颜色，提供整数（0~255）或百分比。例如，白色为 `rgb(255, 255, 255)` 或 `rgb(100%, 100%, 100%)`。此外，我们还能访问一些指定的颜色，比如 `purple`、`lime`、`aqua` 和 `red` 等。颜色关键字列表在 CSS3 颜色模块¹中已得以扩展，包含另外 147 个关键字颜色（它们一般会受到良好支持），CSS3 也为我们提供了其他许多选项：HSL、HSLA 和 RGBA。这些新颜色类型的最明显更改是声明半透明颜色的能力。

6.3.1 RGBA

RGBA 的工作原理类似于 GRB，但它添加了第 4 个值：`alpha`，也就是不透明度级别。前 3 个值仍然表示红色、绿色和蓝色。对于 `alpha` 值，1 表示完全不透明，0 表示完全透明，0.5 表示 50% 不透明。可以使用 0 到 1 之间的任何值，包含 0 和 1。

RGB 也可以使用十六进制表示法表示为 #RRGGBB，与此不同，RGBA 没有十六进制表示法。人们探讨过包含一个针对 RGBA 的 8 字符十六进制值，比如 #RRGGBBAA，但这还有待添加到规范草案中。

例如，让我们看一下注册表单。我们希望该表单具有较暗的颜色，同时仍然保留网站背景的颗粒状纹理。为此，我们将使用 RGBA 颜色 0,0,0,0.2，换句话说，就是 80% 透明的纯黑色：

css/styles.css (excerpt)

```
form {
  :
  background: rgba(0,0,0,0.2) url(../images/bg-form.png) no-repeat
  ↪bottom center;
}
```

因为 Internet Explorer 8 和更低版本缺乏对 RGBA 的支持，所以要声明 RGBA 颜色，请确保在它前面添加了 Internet Explorer 可以理解的颜色。Internet Explorer 将呈现它能够理解的最后一个颜色，所以它将跳过 RGBA 颜色。其他浏览器同时能够理解两种颜色，但是 CSS 级联，所以当遇到 RGBA 颜色时，它们会使用 RGBA 颜色覆盖 Internet Explorer 颜色。

在上面的示例中，我们实际上能够很好地处理没有背景颜色的旧版 Internet

¹ <http://www.w3.org/TR/css3-color/>

Explorer，因为我们使用的颜色几乎是透明的。

6.3.2 HSL 和 HSLA

HSL 表示色调、饱和度和亮度。使用 RGB 时，需要相应地更改所有 3 种颜色值来操作颜色的饱和度或亮度，与此不同，使用 HSL，可以仅调整饱和度或亮度，保持基本的色调不变。HSL 语法使用整数表示色调，使用百分比值表示饱和度和亮度¹。

虽然显示器以 RGB 形式显示颜色，但是浏览器可以将所提供的 HSL 值转换为显示器可显示的颜色。

hsl() 声明接受以下 3 种值。

- 色调表示为 0 到 359 度。例如：0=红色、60=黄色、120=绿色、180=青色、240=蓝色，300=洋红色。当然，可以自由使用它们之间的任何值。
- 饱和度表示为百分比。100%是标准的饱和度值。饱和度 100%将为全色调；饱和度 0 将为一个灰色阴影，会忽略色调值。
- 亮度用百分比表示，标准值为 50%。亮度 100%将为白色，50%将为实际的色调，0%将为黑色。

HSL 还可以包含一个不透明度值。例如，hsla(300, 100%, 50%, 0.5) 是拥有全饱和度和标准亮度，以及 50%不透明度的洋红色。

HSL 模拟人类的眼睛感知颜色的方式，所以设计人员可能更容易理解，而且如前所述，它可以更快速、轻松地调整。可以自由使用您最熟悉的语法，但请记住，如果需要支持 Internet Explorer 8 或更低版本，通常需要使用十六进制表示法。

我们总结一下在 CSS 中编写颜色的所有方式，深红色可写为：

```
#800000;
maroon;
rgb(128,0,0);
rgba(128,0,0,1.0);
```

¹ 颜色理论的全面介绍，以及饱和度和亮度的含义不属于本书的介绍范围。如果希望了解更多信息，那么可以阅读 Jason Beaird 的 *The Principles of Beautiful Web Design* (SitePoint: 墨尔本, 2010 年) [<http://www.sitepoint.com/books/design2/>]，其中包含对颜色的出色介绍


```

❏ hsl(0,100%,13%);
❏ hsla(0,100%,13%,1.0)。

```

6.3.3 不透明度

除了使用 HSLA 和 RGBA 颜色指定透明度，CSS3 还为我们提供了 `opacity` 属性。`opacity` 设置在其上声明它的元素的不透明度。类似于 `alpha` 透明度，不透明度值是 0 到 1（含）之间的浮点值。不透明度值 0 将元素定义为完全透明，而不透明度值 1 表示元素完全不透明。

我们看一个示例：

```

div.halfopaque {
    background-color: rgb(0, 0, 0);
    opacity: 0.5;
    color: #000000;
}

div.halfalpha{
    background-color: rgba(0, 0, 0, 0.5);
    color: #000000;
}

```

虽然乍看起来上面的两个声明块是相同的，但是它们之间实际上存在一个重要区别。`opacity` 设置元素及其所有子元素的不透明度值，但是半透明的 RGBA 或 HSLA 颜色只会影响在其上声明它的元素，对其他元素没有影响。

在上面的示例中，`halfopaque div` 中的任何文本也都将为 50% 不透明（很可能使得文本难以阅读！）；但是 `halfopaque div` 上的文本将仍然为 100% 不透明。

所以，虽然 `halfopaque div` 属性是创建半透明元素快速、轻松的解决方案，但是仍然应该留意这一后果。

6.4 实际应用

现在我们已经介绍了所有可用的 CSS 选择器和新的颜色内容，可以真正开始设置样式了。

对于本章后续部分，我们将设置 The HTML5 Herald 头版中一小部分的样式，这将演示如何添加圆角、文本阴影和方框阴影。

在 The HTML5 Herald 头版的右侧边栏中是一系列不断变化的广告，我们将它们标记为了 `aside` 内的 `article` 元素（参见第 2 章）。第一个广告是一个旧的“Wanted”海报样式广告，提醒读者留意强大且危险的 HTML5 和 CSS3。该广告的最终外观如图 6.1 所示。



图 6.1 我们的“Wanted”广告

您将注意到，广告中央的深灰色方框拥有一个双重边框，它具有圆角以及一种 3 维的“立体”效果。显示为“<HTML5> & {CSS3}”的文本也拥有一个区别于背景的阴影。凭借 CSS3，所有这些效果都可使用一些简单代码来实现，无需依靠图像或 JavaScript。下面让我们看看这是如何完成的。

方框的标记为 `<HTML5> & {CSS3}`。与 HTML 实体不同，它非常直观！

在向它应用任何样式之前，我们需要选择它。当然，可以仅向标记添加一个 `class` 属性，但这样做没有意义。我们这里是为了了解 CSS3，所以应该尝试和使用一些新奇的选择器。

我们的方框不是页面上的唯一一个 `a` 元素，但它可能是边栏中紧挨一段之后的唯一一个 `a` 元素。在这种情况下，最好挑选出该方框。我们也知道添加 CSS3 之前的一些样式的基本知识，那么让我们开始操作吧！

css/styles.css (excerpt)

```
aside p + a {
  display: block;
  text-decoration: none;
  border: 5px double;
  color: #ffffff;
  background-color: #484848;
  text-align: center;

  font-size: 28px;
  margin: 5px 5px 9px 5px;
  padding: 15px 0;
  position: relative;
}
```

效果还不错！如图 6.2 所示，我们获得了想要的外观（可以在早期浏览器上看到）。这也是 Internet Explorer 8 和更低版本显示的外观，但字体样式（我们将在第 9 章中添加）不同。

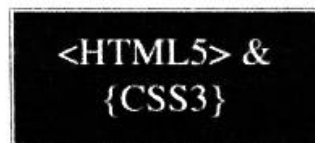


图 6.2 广告链接的基本外观

回想一下，Internet Explorer 6 缺乏对相邻兄弟选择器的支持，所以如果需要向该浏览器提供支持，可以使用更常见的 id 或 class 选择器。

此演示效果很好，应该可接受，网页无需在所有浏览器中看起来都一样。拥有旧版 Internet Explorer 的用户不会知道他们缺少了一些东西，但是我们仍然可为更优秀的浏览器提供支持。下面让我们更进一步，添加一些改进。

6.5 圆角：border-radius

border-radius 属性可用于创建圆角，而无需图像或其他标记。为了向方框添加圆角，我们需要添加以下代码：

```
-moz-border-radius: 25px;
border-radius: 25px;
```

Safari、Chrome、Opera、Internet Explorer 9 和 Firefox 4 支持没有供应商前缀的圆角（单独的 border-radius）。对于 Firefox 3.6 及更早版本，我们仍然需要包含添加了供应商前缀的 -moz-border-radius，但在阅读本文时，这些版本可能很老了，没有必要为它们提供支持。

图 6.3 显示了添加了这些属性后链接的外观。

border-radius 属性实际上是一种短写法。对于我们的 a 元素，所有角都具有相同大小且是对称的。如果想要不同大小的角，可以声明最多 4 个不同的值，例如 border-radius: 5px 10px 15px 20px;。类似于 padding、margin 和 border，可以单独调整每个值：

```
border-top-left-radius: 5px;
border-top-right-radius: 10px;
border-bottom-right-radius: 15px;
border-bottom-left-radius: 40px;
```

针对旧版 Firefox 的 -moz- 前缀形式使用稍微不同的语法：


```
-moz-border-radius-topleft: 5px;
-moz-border-radius-topright: 10px;
-moz-border-radius-bottomright: 15px;
-moz-border-radius-bottomleft: 40px;
```

最终得到的奇怪方框如图 6.4 所示。



图 6.3 向链接添加圆角



图 6.4 可以单独设置每个角的半径

当使用短写的 border-radius 时,角的顺序为左上角、右上角、右下角和左下角。也可以仅声明两个值,在这种情况下,第一个值用于左上角和右下角,第二个值用于右上角和左下角。声明 3 个值时,第一个值表示左上角,第二个值设置右上角和左下角,第三个值表示右下角。

我们建议使用短写法,因为它短得多,而且在旧版 Firefox 不再需要支持之后,它可以避免使用两种不同语法的需要。

也可以创建不对称的角,在每一边使用不同的半径。不是圆形,这些角将是椭圆形的。如果为 4 个长写的值中的每一个都提供两个值,可以分别定义 1/4 个椭圆的水平半径和垂直半径。例如, border-bottom-left-radius: 20px 10px; 将创建一个椭圆形的左下角。

当为椭圆角使用短写法时,使用斜杠将水平和垂直半径分开。border-radius: 20px / 10px; 将创建 4 个相同的椭圆角, border-radius: 5px 10px 15px 20px / 10px 20px 30px 40px; 将创建 4 个不同的椭圆角。最后一个示例将创建如图 6.5 所示的角。有趣吧? 是的。漂亮吗? 不是很漂亮。

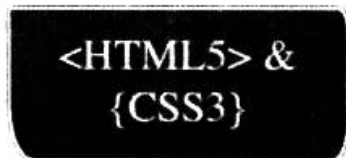


图 6.5 4 个有趣、不同的椭圆角



长远思考

当包含带有前缀的属性时,后面始终附加正确书写、没有前缀、符合标准的语法。这将确保网站向前兼容!

The HTML5 Herald 中还有另外一个元素使用了圆角：注册表单的提交按钮。现在让我们将这些角设置为圆角：

css/styles.ss (excerpt)

```
input[type=submit] {
  -moz-border-radius: 10%;
  border-radius: 10%;
}
```

可以在上面的 CSS 中注意到两点：我们使用了一个属性选择器来定位 submit 输入类型，我们还使用了百分比代替圆角的像素值。如果需要在以后向网站添加更多表单，这将带来很大方便，其他提交按钮可能比注册页面上的按钮小，而且通过使用百分比，圆角将随按钮大小按比例缩放。

border-radius 属性可应用于所有元素，但当 border-collapse 属性设置为 collapse 时，它无法用于 table 元素。



如何处理早期浏览器？

一般而言，无需在早期浏览器中提供相同的外观，但有时客户可能会坚持要求这么做。对于圆角，一种常用方法是动态生成 4 个额外的元素——每个元素对应一个角。然后使用 JavaScript 将 4 个 span 添加到希望圆角化的所有元素，在 CSS 中向与相关角对应的每个 span 提供背景图像。

虽然这样的方法提供了所需的外观，但是它们需要 JavaScript、其他标记、CSS 和/或图像。此外，如果发生设计变更（例如颜色、半径或边框更改），将需要重新创建背景图像。幸运的是，一些 JavaScript 解决方案向旧版 Internet Explorer 提供了 CSS3 修饰符，无需额外的图像或标记，CSS3 PIE¹就是一个值得研究的解决方案。

6.6 投影

CSS3 提供了使用 box-shadow 属性向元素添加投影的能力。此属性可用于指定元素上的一个或多个内部和/或外部投影的颜色、高度、宽度、模糊和偏移。

¹ <http://css3pie.com/>

我们通常将投影视为一种效果，它使元素看起来像“悬浮”在页面上方并投下阴影一样；但是，借助对所有这些变量如此细粒度的控制，可以发挥更大创意。对于我们的广告链接，我们可以使用一个没有模糊的 `box-shadow` 来创建 3D 方框的外观。

`box-shadow` 属性接受一个以逗号分隔的投影列表作为它的值。每个阴影使用 2 到 4 个大小值、一种颜色和表示内嵌或内部阴影的关键词 `inset` 来定义。如果没有指定 `inset`，那么默认将阴影绘制在元素外部。

让我们看看在元素上使用的阴影，逐个分析一下每个值的用途：

css/styles.css (excerpt)

```
-webkit-box-shadow: 2px 5px 0 0 rgba(72,72,72,1);
-moz-box-shadow: 2px 5px 0 0 rgba(72,72,72,1);
box-shadow: 2px 5px 0 0 rgba(72,72,72,1);
```

第一个值是水平偏移。正值将创建投向元素右侧的阴影，负值创建投在左侧的阴影。在我们的示例中，阴影投在 `a` 右侧 2 像素处。

第二个值是垂直偏移。正值将阴影放在下方，在元素底部创建阴影。负值将阴影放在上方。在我们的示例中，阴影位于 `a` 下方 5 像素处。

第三个值（如果已包含）是阴影的模糊距离。该值越大，阴影就越模糊。只允许输入正值。我们的阴影没有模糊化，所以我们可以包含值 0 或省略该值。

第四个值确定阴影的扩张距离。正值将创建向所有方向展开的阴影形状。负值缩小阴影。我们的阴影没有扩张，所以我们同样可以包含值 0 或省略该值。

上面的第 5 个值为颜色。您通常希望声明阴影的颜色。如果省略了它，规范表明它默认设置为与元素的 `color` 属性相同。Opera 和 Firefox 支持这种默认行为，但 WebKit 不支持，所以请确保包含了颜色。在上面的示例中，我们使用了一种 GRBA 颜色。在这项具体设计中，阴影为纯色，所以可以仅使用十六进制值。但是在大部分时间里，阴影将是部分透明的，所以通常会使用 RGBA 或 HSLA。

这些声明创建的投影如图 6.6 所示。

默认情况下，阴影就是投影——投射在方框外部。可以在阴影声明开头添加关键字 `inset`，创建内嵌阴影。



图 6.6 向方框添加一个投影以产生深度的错觉

Opera、Firefox 4 和 Internet Explorer 9 支持不带前缀的语法。我们仍然为 Firefox 3.6 及更早版本包含了 `-moz-` 前缀，为 Safari 和 Chrome 包含了 `-webkit-` 前缀。但是，WebKit 的最新开发版本支持不带前缀的版本，Firefox 4 也很快会取代旧版本，所以添加前缀的需要应该会逐渐减少。



Internet Explorer 6 及更高版本上的投影

要在 Internet Explorer 6 到 Internet Explorer 8 中包含投影，必须使用一个类似如下所示的专门滤镜。但是，请注意，几乎不可能使它看起来与 CSS3 阴影完全一样。还应该注意，滤镜对性能具有重大影响，所以应该仅在旧浏览器必须使用它们才能查看阴影时使用它们。此外，这些样式应该位于一个针对较早 Internet Explorer 版本的独立样式表中并附带条件注释，否则它们会使 Internet Explorer 9 上的标准 CSS3 阴影变得杂乱。

```
filter: shadow(color=#484848, direction=220, Strength=8);
filter: progid:DXImageTransform.Microsoft.dropshadow(OffX=2,
    ↳OffY=5, Color='#484848', Positive='true');
```



非矩形阴影？

投影在矩形元素上显示效果不错，包括像我们的元素一样具有圆角的元素。我们在元素上使用了 `border-radius` 属性，所以阴影将沿着角的曲线显示，这看起来效果不错。

但是，请记住，阴影沿着元素的边缘显示，而不是沿着内容的像素显示。所以，如果尝试在半透明图像上使用投影，将会得到奇怪的结果：阴影沿着图像的矩形边框而不是图像内容的轮廓显示。

要在元素上包含多个方框阴影，可以定义一个逗号分隔的阴影列表。当指定了多个阴影时，阴影会从前向后分层，就像浏览器首先绘制最后一个阴影，然后将前一个阴影绘制到它之上。

类似于元素的轮廓，方框阴影在方框模型方面应该是不可见的。换句话说，它们应该对页面的布局没有影响——如果有必要，它们将覆盖其他方框和它们的阴影。我们说“应该”是因为一些浏览器中存在错误，但这样的浏览器很少并且错误可能很快就会修复。

内嵌和多个阴影

The HTML5 Herald 的注册表单拥有一种类似于围绕边缘的渐变背景的效果，但这实际上是一些内嵌方框阴影。

要创建内嵌方框阴影，可以将 `inset` 关键字添加到声明中。在我们的示例中，我们必须包含两个阴影，以便覆盖所有 4 个角：一个阴影针对左上角，另一个针对右下角：

```
css/styles.css (excerpt)

form {
  -webkit-box-shadow:
    inset 1px 1px 84px rgba(0,0,0,0.24),
    inset -1px -1px 84px rgba(0,0,0,0.24);
  -moz-box-shadow:
    inset 1px 1px 84px rgba(0,0,0,0.24),
    inset -1px -1px 84px rgba(0,0,0,0.24);
  box-shadow:
    inset 1px 1px 84px rgba(0,0,0,0.24),
    inset -1px -1px 84px rgba(0,0,0,0.24);
}
```

可以看到，要向一个元素添加多个阴影，只需重复相同的语法（以逗号分隔）。



WebKit 和内嵌阴影

基于 WebKit 的浏览器的最新版本在呈现具有较大 `blur` 值的内嵌方框阴影（比如我们在 The HTML5 Herald 的注册表单上使用的阴影）时，性能会显著下降。

因为 WebKit 同时支持带有 `-webkit-` 前缀和不带前缀的 `box-shadow` 属性形式，所以我们必须从完成的 CSS 中省略它们。只能包含带有 `-moz-` 前缀的属性，所以很不幸，只有 Firefox 能从我们漂亮的较大内嵌阴影获益。

这个错误已在 WebKit 引擎的最新开发版本中得以修复，但将修复应用到每个基于 WebKit 的浏览器的发行版中可能还需要一段时间。因此，如果使用内嵌阴影，请确保进行了大量浏览器测试。

6.7 文本阴影

`box-shadow` 可用于向方框添加阴影, `text-shadow` 可用于向文本节点中各个字符添加阴影。`text-shadow` 是在 CSS2 中新增的, 从 Safari 的第一版开始就受到了它的支持, 现在受到了除 Internet Explorer 9 外的所有最新浏览器版本的支持。

`text-shadow` 属性的语法非常类似于 `box-shadow`, 包括前缀、偏移和添加多个阴影的能力, 但没有扩张, 不允许嵌套阴影:

```
/* single shadow */
text-shadow: topOffset leftOffset blurRadius color;

/* multiple shadows */
text-shadow: topOffset1 leftOffset1 blurRadius1 color1,
             topOffset2 leftOffset2 blurRadius2 color2,
             topOffset3 leftOffset3 blurRadius3 color3;
```

类似于 `box-shadow`, 当声明多个阴影时, 它们会从前往后绘制, 第一个阴影位于最顶部。文本阴影显示在文本本身后面。如果一个阴影太大, 以至于碰到了另一个字母, 它将继续放在该字符后。

我们的文本具有投射到右下角的半透明阴影:

```
css/styles.css (excerpt)

text-shadow: 3px 3px 1px rgba(0, 0, 0, 0.5);
```

这表明阴影向文本下方扩展 3 像素, 向文本右侧扩展 3 像素, 稍微有点模糊 (1 像素), 而且使用具有 50% 不透明度的黑色作为基础颜色。

设置好样式后, 我们的广告链接就快完成了, 如图 6.7 所示。最后一步 (自定义字体) 将在第 9 章中执行。

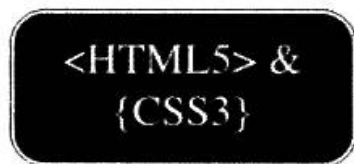


图 6.7 我们的广告链接看起来很漂亮

6.8 更多阴影

我们现在知道了如何在块级元素和文本节点上创建投影。但目前为止, 我们仅

设置了页面一部分的样式：事实上只是一个广告中的一个链接。在继续之前，让我们来创建剩余的阴影。

回头看看网站设计，可以看到，页面上的所有 h1 元素都是大写的并具有投影。文本为深灰色，在右下方具有细微的纯白色投影，提供了一定的深度效果¹。网站顶部的标志行也有一个投影，但它全部采用小写。而文章的标志行没有投影。

我们从“CSS3 选择器”一节中了解到，无需使用类即可定义所有这些元素。下面让我们不使用任何其他标记来定位这些元素：

```
css/styles.css (excerpt)

h1, h2 {
  text-transform: uppercase;
  text-shadow: 1px 1px #FFFFFF;
}
:not(article) > header h2 {
  text-transform: lowercase;
  text-shadow: 1px 1px #FFFFFF;
}
```

第一个声明定位页面上的所有 h1 元素和 h2 元素。第二个声明针对一个 header 中的所有 h2 元素，但前提是该 header 没有嵌套在一个 article 元素中。

我们的文本阴影为纯白色，所以无需使用 alpha 透明颜色或模糊半径。

6.9 小结

我们已掌握了阴影和圆角，是时候从 CSS3 中获得一些乐趣了。下一章将介绍 CSS3 渐变和多个背景图像。

¹ 请访问 <http://twitter.com/#!/themaninblue/status/27210719975964673>

CSS3 渐变和多背景

第 6 章介绍了一些向页面添加装饰性样式（比如阴影和圆角）的方式，这些方式无需使用额外的标记或图像。下一个最常添加到网站且在过去需要使用图像的功能是渐变。CSS3 为我们提供了创建原生的径向和线性渐变的能力，以及在任何元素上包含多个背景图像的能力。使用 CSS3，无需像过去多年那样创建众多的 JPEG 图像，或向我们的标记添加非语义挂钩。

对渐变和多背景的浏览器支持仍在发展，但在阅读本章时，可能已开发出一种方式来支持所有主流浏览器的最新版本，包括 Internet Explorer 9。

我们将首先介绍 CSS3 渐变，但什么是渐变呢？渐变是两种或多种指定颜色之间的平滑过渡。在创建渐变的过程中，可以指定多个中间颜色值，称为色标。每个色标包含一种颜色和一个位置，浏览器从每个色标的颜色淡出到下一个，以创建平滑的渐变。渐变可应用于任何可使用背景图像的地方。这意味着在 CSS 中，渐变在理论上可在任何可使用 `url()` 值的地方采用，比如 `background-image`、`border-image` 以及甚至 `list-style-type`，但目前最一致的支持是对背景图像的支持。

通过使用 CSS 渐变取代图像，可以避免迫使用户下载额外的图像，对灵活布局的支持得到了改进，缩放不再像图像一样像素化。

目前有两种类型的渐变可用于 CSS3：线性和径向。我们将依次介绍它们。

7.1 线性渐变

在线性渐变过程中，颜色沿一条直线过渡：从顶部到底部、从左侧到右侧，或沿任何任意轴。如果使用过 Photoshop 和 Fireworks 等图像编辑工具，应该很熟悉线性渐变，作为复习，图 7.1 给出了一些示例。

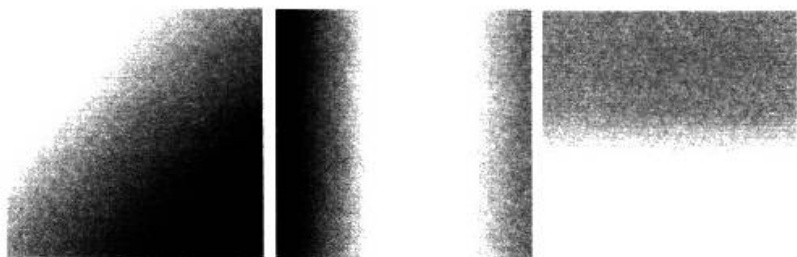


图 7.1 线性渐变示例

类似于图像编辑程序，要创建线性渐变，需要指定一个方向、起始颜色、结束颜色，以及希望沿这条线添加的任何色标。浏览器负责剩余工作，通过绘制与渐变线垂直的颜色线来填充整个元素。它生成从一种颜色到另一种颜色的平滑淡出，沿所指定的方向渐变。

对于浏览器和线性渐变，事情稍微有点杂乱。WebKit 于多年前率先使用一种具体且许多人认为复杂的语法来引入了渐变。在这之后，Mozilla 使用一种更简单且更直观的语法实现了渐变。随后，在 2011 年 1 月，W3C 在 CSS3 中包含了一个建议语法。新语法非常类似于 Firefox 的现有实现——实际上，它们是如此的相近，以至于任何使用新语法编写的渐变都能够很好地应用于 Firefox。W3C 语法已被 WebKit 采用，但是在编写本书时它还处于每日构建阶段，还未引入到 Chrome 和 Safari 中；它们仍在使用旧样式语法。出于向后兼容性原因，这些浏览器将继续支持旧语法，即使在标准表单实现之后。此外，Opera（已发布 11.10 版）支持新的 W3C 线性渐变标准。所有最新的实现都使用了供应商前缀（-webkit-、-moz-和-o-）。



WebKit 每日构建版本

位于 Chrome 和 Safari 核心的 WebKit 引擎是作为 <http://www.webkit.org/> 上的一个开源项目独立存在的。但是，在 WebKit 中实现的新功能经历一定的时间后才会发布在 Chrome 或 Safari 中。在此过程中，仍然可以通过安装一个每日构建版本来测试这些功能，这样称呼是因为它们每天都会构建和发布，整合来自社区

每天工作中的新功能或代码更改。因为它们在开发期间频繁地发布，所以它们可能包含不完整的功能或错误，而且常常不稳定。但是，如果希望测试将整合到 Chrome 或 Safari 中的新功能（比如 W3C 渐变语法），请访问 <http://nightly.webkit.org/>，获取针对 Mac 或 Windows 的 WebKit 每日构建版本。

这仍然为我们留下了一个问题，如何在 IE 和早期版本的 Opera 中处理渐变？幸运的是，Internet Explorer 及 Opera 11.01 和更早版本支持 SVG 背景，而且在 SVG 中创建渐变非常简单。（第 11 章将更详细地介绍 SVG。）最后，所有 Internet Explorer 版本都支持一种专用滤镜，能够创建基本的线性渐变。

不知所措了？不要这样。虽然了解渐变很重要，但是没有必要记住所有浏览器语法。我们将介绍新语法，以及很快就可以忘记的旧样式 WebKit 语法，然后再透露一个小秘密：一些工具可以创建所有需要的样式，所以无需记住每种语法的所有细节。让我们开始吧。

The HTML5 Herald 中有一种线性渐变，位于如图 7.2 所示的第二个广告块中（这正好是推广本书的广告！）。可以注意到，渐变首先从顶部的深色开始，变亮，然后再次变暗，就像为骑自行车的人建立一条道路，最后再次变亮。



图 7.2 The HTML5 Herald 中的线性渐变

为了为我们的广告创建跨浏览器渐变，我们将开始使用新的标准语法。该语法最简单也最容易理解，可能是您未来几年需要使用的唯一语法。在这之后，我们将看看较老的 WebKit 和 Firefox 语法与它有何不同。

7.1.1 W3C 语法

以下是线性渐变的基本语法：

```
background-image: linear-gradient( ... );
```

在括号内，可以指定渐变的方向，然后提供一些色标。对于方向，可以提供一个角度，渐变应该沿着它发生，或者开始渐变的一边或角——在这种情况下，它将朝着对边或对角渐变。对于角度，使用以度（deg）为单位的值。0deg 指向右侧，90deg 向上，沿逆时针方向依此类推。对于边或角，使用 top、bottom、left 和 right 关键字。指定方向后，提供色标，色标由一种颜色和一个百分比或长度组成，后者指定渐变到该色标的距离。

需要了解的内容很多，让我们看一些渐变示例。为了方便说明，我们将使用一种仅有两个色标的渐变：从#FFF（白色）到#000（黑色）。

为了让渐变从元素顶部向底部变化，如图 7.3 所示，可以指定以下任何方式（我们的示例使用了-moz-前缀，但请记住，-webkit-和-o-支持相同的语法）：

```
background-image: -moz-linear-gradient(270deg, #FFF 0%, #000 100%);
background-image: -moz-linear-gradient(top, #FFF 0%, #000 100%);
background-image: -moz-linear-gradient(#FFF 0%, #000 100%);
```

最后一个声明之所以有效，是因为 top 是在没有指定方向时的默认设置。

因为已假设第一个色标为 0%，最后一个色标为 100%，所以在该示例中省略百分比也能实现相同的结果：

```
background-image: -moz-linear-gradient(#FFF, #000);
```

现在，让我们沿着一个角度渐变，添加另一个色标。假设我们希望从黑色变为白色，然后再从白色变为黑色：

```
background-image: -moz-linear-gradient(30deg, #000, #FFF 75%, #000);
```

我们在渐变路线上放置色标 75%，所以相对于渐变的起点，白色区域更接近渐变的终点，如图 7.4 所示。

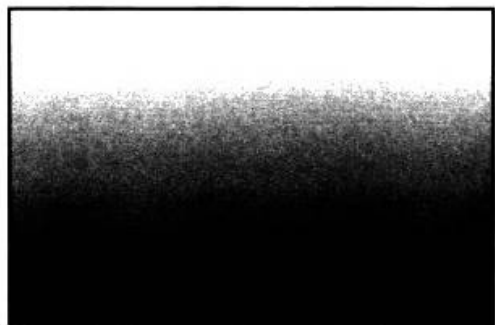


图 7.3 从元素顶部中央到底部中央的白色到黑色渐变



图 7.4 包含 3 个色标的渐变

可以将第一个色标放在除 0%外的其他某个位置，将最后一个色标放在除 100%外的某个位置。0%到第一个色标之间的所有空间都将具有与第一个色标相同的颜色，最后一个色标与 100%之间的所有空间都将具有与最后一个色标相同的颜色。下面是一个示例：

```
background-image: -moz-linear-gradient(30deg, #000 50%, #FFF 75%,
    ➡#000 90%);
```


最终的渐变结果如图 7.5 所示。

实际上并不需要指定任何色标的位置。如果省略它们，它们将均匀地分布。下面是一个示例：

```
background-image:
  -moz-linear-gradient(45deg,
    #FF0000 0%,
    #FF6633 20%,
    #FFFF00 40%,
    #00FF00 60%,
    #0000FF 80%,
    #AA00AA 100%);
```

```
background-image:
  -moz-linear-gradient(45deg,
    #FF0000,
    #FF6633,
    #FFFF00,
    #00FF00,
    #0000FF,
    #AA00AA);
```

上面的每一个声明都产生了呈一定角度的单调彩虹效果。请注意，我们添加了换行和缩进，以易于阅读——它们不是必要的。

颜色从一个色标平滑地过渡到下一个。但是，如果两个色标位于渐变方向上的相同位置，颜色将不会淡出，而会生硬地开始和停止。这是一种创建条带状背景效果的方式，如图 7.6 所示。



图 7.5 通过偏移开始色标和结束色标而限制到一个窄区域的渐变



图 7.6 恰当地放置色标可创建条带状背景

以下是用于构造该示例的样式：

```
background-image:
  -moz-linear-gradient(45deg,
    #000000 30%,
    #666666 30%,
    #666666 60%,
    #CCCCCC 60%,
    #CCCCCC 90%);
```

预计在近期，这个更新的语法版本是您需要编写的唯一语法，但现在还没到这一时刻。

7.1.2 旧 WebKit 语法

如前所述，WebKit 的最新开发版本采用了 W3C 语法，但是基于 WebKit 的浏览器的最新版本仍在使用较旧的语法，该语法稍微复杂一些。因为这些浏览器仍然需要受到支持，让我们快速看一下旧语法，再次使用第一个白色到黑色渐变的示例：

```
background-image:
  -webkit-gradient(linear, 0% 0%, 0% 100%, from(#FFFFFF),
    to(#000000));
```

没有使用特定的 `linear-gradient` 属性，有一个通用的 `-webkit-gradient` 属性，可以使用它指定渐变类型（在本例中为线性）作为第一个参数。然后线性渐变需要起点和终点来确定渐变的方向。起点和终点可使用百分比、数字值或关键字 `top`、`bottom`、`left`、`right` 或 `center` 来指定。

下一步是声明渐变的色标。可以使用 `from` 关键字包含起始颜色，使用 `to` 关键字包含终止颜色。然后，可以包含任意数量的中间颜色，使用 `color-stop()` 函数创建色标。`color-stop()` 函数的第一个参数是色标的位置（表示为百分比），第二个参数是该位置的颜色。

下面是一个示例：

```
background-image:
  -webkit-gradient(linear, left top, right bottom,
    from(red),
    to(purple),
    color-stop(20%, orange),
    color-stop(40%, yellow),
    color-stop(60%, green),
    color-stop(80%, blue));
```

使用这段代码，我们重新创建了呈一定角度的彩虹，这使人想起了大约 1996 年的 GeoCities。

实际上没有必要使用 `from` 和 `to` 指定开始和结束颜色。因为 `from(red)` 等效于 `color-stop(0, red)`，我们也可以这样编写：

```
background-image:
  -webkit-gradient(linear, left top, right bottom,
    color-stop(0, red),
    color-stop(20%, orange),
    color-stop(40%, yellow),
    color-stop(60%, green),
    color-stop(80%, blue),
    color-stop(100%, purple));
```

如果不声明 `from` 或 0% 的色标，第一个色标的颜色将用于直到第一个色标的所有区域。元素从容器边缘到第一个指定的色标的区域将具有所声明的纯色，颜色会在第一个色标转变为第二个色标的颜色。在最后一个色标及以后，将使用最后一个色标的颜色。换句话说，如果第一个色标位于 40% 处，最后一个色标位于 60% 处，那么第一种颜色将用在从 0% 到 40% 的区域，最后一种颜色将在 60% 到 100% 的区域内显示，40% 到 60% 的区域是两种颜色之间的渐变。

可以看到，这比 Mozilla 语法更加复杂。幸运的是，可以使用工具来自动为指定渐变生成所有需要的代码。本节末将介绍其中一些工具，但是首先我们将看看如何使用两种语法为 The HTML5 Herald 创建跨浏览器渐变。好消息是，因为旧 WebKit 语法使用了不同的属性名称（`-webkit-gradient`，而不是 `-webkit-linear-gradient`），所以可以并列使用两种语法而不会发生冲突。事实上，较新的 WebKit 中仍然支持旧语法，所以浏览器将使用最后声明的语法。

7.1.3 实际应用

现在我们很好地理解了如何声明线性渐变，下面让我们来声明自己的渐变。

如果设计人员在设计中包含了渐变，那么它很可能是在 Photoshop 或另一个图像编辑程序中创建的。使用此方式有一定的好处：如果拥有原始文件，就可以轻松地再现设计人员想要的效果。

打开 Photoshop 并检查我们希望用于广告的渐变（如图 7.7 所示），可以看到我们的渐变是线性的，具有 5 个色标，它们仅仅更改一种颜色（黑色）的不透明度。



图 7.7 Photoshop 中的示例线性渐变

通过 Photoshop 屏幕截图可以注意到，颜色从 40%的不透明度开始，第一个色标的位置位于 37%，其不透明度为 0%。我们可以使用此工具从 CSS 声明中捕捉数据，首先是针对 Firefox、Opera 11.10 和较新的 WebKit 浏览器的 W3C 语法声明：

css/styles.css (excerpt)

```
#ad2 {
  :
  background-image:
    -moz-linear-gradient(
      270deg,
      rgba(0,0,0,0.4) 0,
      rgba(0,0,0,0) 37%,
      rgba(0,0,0,0) 83%,
      rgba(0,0,0,0.06) 92%,
      rgba(0,0,0,0) 98%
    );
  background-image:
    -webkit-linear-gradient(
      270deg,
      rgba(0,0,0,0.4) 0,
      rgba(0,0,0,0) 37%,
      rgba(0,0,0,0) 83%,
      rgba(0,0,0,0.06) 92%,
      rgba(0,0,0,0) 98%
    );
  background-image:
    -o-linear-gradient(
      270deg,
      rgba(0,0,0,0.4) 0,
      rgba(0,0,0,0) 37%,
      rgba(0,0,0,0) 83%,
      rgba(0,0,0,0.06) 92%,
      rgba(0,0,0,0) 98%
    );
}
```

我们希望渐变从广告最顶部过渡到底部，所以我们将角度设置为 270deg（向底部渐变）。然后我们添加来自 Photoshop 渐变的所有色标。请注意，我们省略了渐变的终点，因为最后一个色标位于 98%——该色标后的所有区域将与该节点具有相同颜色（在本例中，不透明度为 0%或完全透明的黑色）。

现在让我们添加旧 WebKit 语法，在最后使用不带前缀的版本来实现不会过时的声明：

css/styles.css (excerpt)

```
#ad2 {
  :
  background-image:
    -webkit-gradient(linear,
      from(rgba(0,0,0,0.4)),
      color-stop(37%, rgba(0,0,0,0)),
      color-stop(83%, rgba(0,0,0,0)),
      color-stop(92%, rgba(0,0,0,0.16)),
      color-stop(98%, rgba(0,0,0,0)));
  background-image:
    -webkit-linear-gradient(
      270deg,
      rgba(0,0,0,0.4) 0,
      rgba(0,0,0,0) 37%,
      rgba(0,0,0,0) 83%,
      rgba(0,0,0,0.06) 92%,
      rgba(0,0,0,0) 98%
    );
  background-image:
    linear-gradient(
      270deg,
      rgba(0,0,0,0.4) 0,
      rgba(0,0,0,0) 37%,
      rgba(0,0,0,0) 83%,
      rgba(0,0,0,0.06) 92%,
      rgba(0,0,0,0) 98%
    );
}
```

我们现在拥有了可在 Mozilla、Opera 和基于 WebKit 的浏览器中正确显示的渐变。

7.1.4 使用 SVG 的线性渐变

我们仍然还有一些浏览器需要添加线性渐变。在 Opera 11.01 和更早的版本中，

以及最重要的 Internet Explorer 9 中，我们可以将 SVG 文件声明为背景图像。通过在 SVG 文件中创建渐变，并将该 SVG 声明为元素的背景图像，我们可以重新创建使用 CSS3 渐变所实现的相同效果。



SV 什么？

SVG 表示可缩放矢量图形。它是一种基于 XML 的语言，使用一组元素定义矢量图形——就像在 HTML 中用于定义文档结构的语言一样。第 11 章将更详细地介绍 SVG，但现在我们将跳过基本知识，因为我们所创建的只是一个简单的渐变。

SVG 文件听起来有点奇怪，但使用它创建渐变非常简单。下面同样是我们的渐变，但采用了 SVG 形式：

images/gradient.svg (excerpt)

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
  "http://www.w3.org/TR/2001/REC-SVG-20050904/DTD/svg10.dtd">
<svg xmlns="http://www.w3.org/2000/svg"
  xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1">
<title>Module Gradient</title>
<defs>
  <linearGradient id="grad" x1="0" y1="0" x2="0" y2="100%">
    <stop offset="0" stop-opacity="0.3" color-stop="#000000" />
    <stop offset="0.37" stop-opacity="0" stop-color="#000000" />
    <stop offset="0.83" stop-opacity="0" stop-color="#000000" />
    <stop offset="0.92" stop-opacity="0.06" stop-color="#000000" />
    <stop offset="0.98" stop-opacity="0" stop-color="#000000" />
  </linearGradient>
</defs>
<rect x="0" y="0" width="100%" height="100%"
  style="fill:url(#grad)" />
</svg>
```

看一下 SVG 文件，应该会注意到它非常类似于 CSS3 中用于线性渐变的语法。我们在 linearGradient 元素中声明渐变类型和方向，然后添加色标。方向使用起点和终点坐标设置，从 (x1, y1) 到 (x2, y2)。色标很容易理解，拥有 0 到 1 之间的一个偏移，以确定它们的位置及其颜色的色标。声明渐变之后，必须创建一个矩形 (rect 元素) 并使用 style 属性向其中填充我们的渐变。

所以，我们创建了一种非常漂亮的细微渐变，但如何在网站上使用它呢？使用 **.svg** 扩展名保存 SVG 文件。然后，在您的 CSS 中，使用相同语法将 SVG 声明为背景图像，就像它是 JPEG、GIF 或 PNG 一样：

```
css/styles.css (excerpt)

#ad2 {
  :
  background-image: url(../images/gradient.svg);
  :
}
```

SVG 背景应该在 CSS3 渐变之前声明，所以能理解二者的浏览器将使用后者。许多浏览器非常智能，甚至在 SVG 被 CSS 中另一个 `background-image` 属性覆盖时都不会下载 SVG。

CSS 线性渐变与 SVG 版本之间的主要区别在于，SVG 背景图像不会像 CSS 渐变一样默认设置为容器的 100% 的高度和宽度。要让 SVG 填充整个容器，可以将 SVG 矩形的 `height` 和 `width` 声明为 100%。

7.1.5 使用 Internet Explorer 滤镜的线性渐变

对于 Internet Explorer 9 以前的版本，我们可以使用专用的 Internet Explorer 滤镜语法来创建简单的渐变。Internet Explorer 渐变滤镜不支持色标、渐变角度或我们稍后将看到的径向渐变。您能做的只是指定渐变是水平的还是垂直的，以及结束颜色和开始颜色。这非常简单，但如果需要在这类较旧的浏览器上实现渐变，它可以提供解决方案。

针对 Internet Explorer 的滤镜语法为：

```
filter:progid:DXImageTransform.Microsoft.gradient(GradientType=0,
➤startColorstr='#COLOR', endColorstr='#COLOR'); /* IE6 & IE7 */
-ms-filter:"progid:DXImageTransform.Microsoft.gradient(GradientType=
➤0,startColorstr='#COLOR', endColorstr='#COLOR')"; /* IE8 */
```

`GradientType` 参数设置为 1 表示水平渐变，设置为 0 表示垂直渐变。

因为我们为广告块使用的渐变需要色标，所以我们将跳过 IE 滤镜。没有渐变的广告看起来仍然不错，所以这没什么问题。



滤镜说明

如前所述, Internet Explorer 的滤镜可能对性能具有重大影响, 所以应该在需要时谨慎地使用它们。计算滤镜效果的显示需要花一定的处理时间, 一些效果显示得比另一些慢。SVG 可拥有类似 (但少得多) 的效果, 所以如果要使用这些备用方案, 请确保在众多的浏览器中测试了您的网站。

7.1.6 便捷的工具

理解了如何创建线性渐变并掌握了这些复杂语法之后, 就可以忘记所学到的几乎所有知识了。有一些非常有用的工具可帮助创建线性渐变, 无需为不同的浏览器语法重新创建代码 4 次。

John Allsop 的 <http://www.westciv.com/tools/gradients/> 就是一个可为 Firefox 和 WebKit 创建包含色标的渐变的工具。请注意, Firefox 和 WebKit, 以及径向和线性渐变都具有独立的选项卡。该工具仅使用十六进制颜色表示法创建渐变, 但它提供了复制并粘贴代码的功能, 所以可以复制它并根据喜好将十六进制颜色值转换为 RGBA 或 HSLA。

Damian Galarza 的 <http://gradients.glrzad.com/> 同时提供了色标和 RGB。它甚至允许使用 HSL 颜色选取器设置颜色, 并在代码中将它转换为 RGB。它没有提供 alpha 透明度, 但因为生成的代码是 RGB 格式的, 所以可以轻松更新。这个渐变生成器比 Westciv 更加强大, 但对新手而言可能比较复杂。

最后, Paul Irish 的 <http://css3please.com/> 可用于创建线性渐变, 但它不支持色标。您可能想知道为什么有必要提及它, 这是因为它是所提及的工具中唯一同时提供了针对 Internet Explorer 的滤镜语法和其他渐变语法的工具。另外, 除了渐变, 它还可以为其他许多功能 (比如阴影和圆角) 提供跨浏览器语法。

7.2 径向渐变

径向渐变是圆形或椭圆形渐变。颜色不再沿一条直线轴变化, 而是从一个起点朝所有方向混合。WebKit 和 Mozilla (从 Firefox 3.6 开始) 支持径向渐变。尽管 Opera 11.10 已开始支持线性渐变, 但它未提供对径向渐变的支持; 与线性渐变一样, 径向渐变也可以在 SVG 中创建, 所以 Opera 和 Internet Explorer 9 也可以提供支持。径

向渐变在 Internet Explorer 8 和更早版本中完全不受支持, 这些版本甚至不支持滤镜。

7.2.1 W3C 语法

我们首先用一个简单的圆形渐变演示一下标准语法:

```
background-image: -moz-radial-gradient(#FFF, #000);
background-image: -moz-radial-gradient(center, #FFF, #000);
background-image: -moz-radial-gradient(center, ellipse cover,
    #FFF, #000);
```

上面 3 句声明是等效的, 都将生成图 7.8 所示的渐变。您至少需要提供一个开始颜色和结束颜色。也可以提供渐变的中心位置作为第一个参数, 提供一个形状和大小作为第二个参数。

接下来看一下位置:

```
background-image: -moz-radial-gradient(30px 30px, #FFF, #000);
```

这会将渐变的中心放在离顶部 30 像素且离元素左侧 30 像素的位置, 如图 7.9 所示。与 background-position 一样, 可以使用值、百分比或关键字来设置渐变的位置。

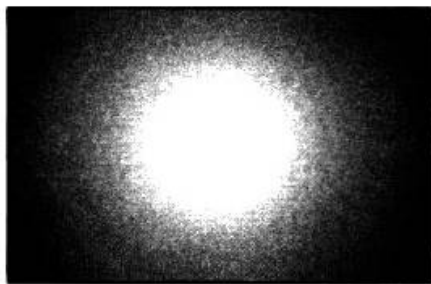


图 7.8 一个简单、居中的径向渐变

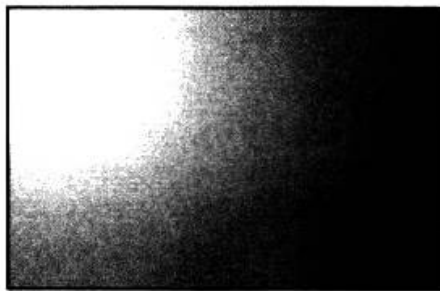


图 7.9 偏离了中心的渐变位置

现在让我们看一下形状和位置参数。形状可接受两个值之一: circle 或 ellipse, 后者是默认设置。

对于大小, 可以使用下列值之一。

closest-side

渐变的形状到达方框离其中心最近的一边 (圆形), 或者同时到达离其中心最近的垂直和水平边 (椭圆形)。

closest-corner

调整渐变的形状大小，使它准确到达方框中离其中心最近的角。

farthest-side

类似于 `closest-side`，但形状的大小设置为到达方框中离其中心最远的边（或在椭圆形情况下，最远的垂直和水平边）。

farthest-corner

设置渐变的形状大小，使它准确到达方框中离其中心最远的角。

contain

与 `closest-side` 同义。

cover

与 `farthest-corner` 同义。

根据规范，也可以提供第二组值来显式定义径向渐变的水平和垂直大小。目前只有 WebKit 支持此方法，但在不久的将来 Firefox 应该会添加支持。但是现在，如果希望所有支持的浏览器中创建相同的渐变，可能应该坚持使用上述约束。

色标语法与线性渐变相同：一个颜色值后跟一个可选的色标位置。下面让我们看最后一个示例：

```
background-image: -moz-radial-gradient(30px 30px, circle
➡farthest-side, #FFF, #000 30%, #FFF);
```

这将创建一种类似图 7.10 的渐变（修改了大小和形状，还具有一个额外的色标）。

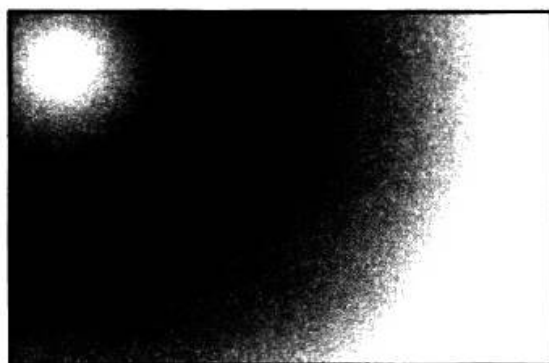


图 7.10 径向渐变

7.2.2 旧 WebKit 语法

要使用 Safari 和 Chrome 目前支持的旧样式的 WebKit 语法创建图 7.10 中的示例，需要按下列形式编写代码：

```
background-image: -webkit-gradient(radial, 30 30, 0, 30 30, 100%,
➡from(#FFFFFF), to(#FFFFFF), color-stop(.3, #000000))
```

为径向渐变使用之前在线性渐变中使用的相同的 `-webkit-gradient` 属性，不同之处在于我们将 `radial` 作为第一个参数。接下来的 4 个参数分别是两个圆的位置和半径，渐变方向为从内部圆到外部圆。容易混淆的是，这些值是使用像素定义的，但没有使用 `px` 单位。也可以百分比形式指定值，在这种情况下需要包含 `%` 符号。您现在应该明白为什么 W3C 选择了 Mozilla 语法版本，而不是此版本了。

此外，内部圆不需要位于外部圆的中心。如果第一个点与第二个点相同，渐变将是对称的，类似于图 7.10 所示。但是，如果两个点不同，内部圆将偏离中心，所以渐变将是不对称的。如果内部圆的中心在外部圆的边界之外，而不是一个圆的中心在另一个圆之内，将得到一种非常个性的三角形渐变效果，如图 7.11 所示。

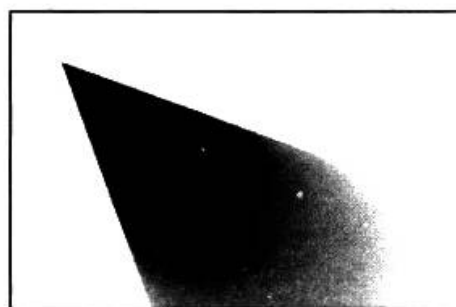


图 7.11 较老的 WebKit 径向渐变语法支持一些有趣的效果

以下是用于创建该渐变的代码：

```
background-image: -webkit-gradient(radial, 200 200, 100, 100 100, 40,
from(#FFFFFF), to(#000000));
```

与线性渐变一样，可以使用 `color-stop` 函数插入更多颜色。色标的语法对于线性和径向渐变是相同的。

您通常希望创建的渐变在较老的 Chrome 和 Safari 版本中看起来与在这些浏览器的较新版本和 Firefox 中相同，所以应该仅限于可使用 W3C 语法再现的渐变类型。但是，如果希望专门为 WebKit 浏览器创建渐变（例如，用于移动平台），了解存在这些额外的选项会很有用。如前所述，在可预见的未来，WebKit 浏览器将继续支持旧语法。

7.2.3 实际应用

让我们利用所学的所有知识为 The HTML5 Herald 实现一种径向渐变。您可能已注意到，表单提交按钮的背景中已有一种径向渐变。该径向渐变的中心位于按钮区域外，朝按钮的左下方变化，如图 7.12 所示。



图 7.12 The HTML5 Herald 注册表单中一个按钮的径向渐变

我们希望声明至少 3 个背景图像：一个 SVG 文件用于 Opera 和 Internet Explorer 9，一个用于 Chrome 和 Safari 较旧的 WebKit 语法，一个针对 Firefox 的 `-moz-` 供应商前缀版本。您也可以声明较新的 WebKit 供应商前缀版本（目前仅在 WebKit 每日构建版本中可用），以及不带前缀的版本：

css/styles.css (excerpt)

```
input[type=submit] {
  :
  background-color: #333;
  /* SVG for IE9 and Opera */
  background-image: url(../images/button-gradient.svg);
  /* Old WebKit */
  background-image: -webkit-gradient(radial,
    30% 120%, 0, 30% 120%, 100,
    color-stop(0,rgba(144,144,144,1)),
    color-stop(1,rgba(72,72,72,1)));
  /* W3C for Mozilla */
  background-image: -moz-radial-gradient(30% 120%, circle,
    rgba(144,144,144,1) 0%,
    rgba(72,72,72,1) 50%);
  /* W3C for new WebKit */
  background-image: -webkit-radial-gradient(30% 120%, circle,
    rgba(144,144,144,1) 0%,
    rgba(72,72,72,1) 50%);
  /* W3C unprefixed */
  background-image: radial-gradient(30% 120%, circle,
    rgba(144,144,144,1) 0%,
    rgba(72,72,72,1) 50%);
}
```


圆的中心位于离左边 30%、离顶边 120%的位置，所以它实际上位于容器底边下方。我们包含了两个色标，分别针对颜色 #484848（或 `rgb(72,72,72)`）和 #909090（或 `rbg(144,144,144)`）。

以下是备用的 SVG 文件，我们这里仅简单解释一下，因为它的语法非常简单并且第 11 章中将会介绍 SVG。

button-gradient.svg (excerpt)

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
  "http://www.w3.org/TR/2001/REC-SVG-20050904/DTD/svg10.dtd">
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="
  http://www.w3.org/1999/xlink" version="1.1">
<title>Button Gradient</title>
<defs>
  <radialGradient id="grad" cx="30%" cy="120%" fx="30%" fy="120%"
    r="50%" gradientUnits="userSpaceOnUse">
    <stop offset="0" stop-color="#909090" />
    <stop offset="1" stop-color="#484848" />
  </radialGradient>
</defs>
<rect x="0" y="0" width="100%" height="100%"
  style="fill:url(#grad)" />
</svg>
```

7.3 重复渐变

有时我们希望创建一种在一个元素的背景上重复的渐变“模式”。虽然可以通过重复背景图像（使用 `background-repeat`）创建线性重复渐变，但是没有创建重复的径向渐变的类似方式。幸运的是，CSS3 通过 `repeating-linear-gradient` 和 `repeating-radial-gradient` 语法提供了补救方法。Firefox 3.6 及更高版本、Safari 5.0.3 及更高版本、Chrome 10 及更高版本和 Opera 11.10 及更高版本支持带供应商前缀的 `repeating-linear-gradient` 语法。

包含 `repeating-linear-gradient` 和 `repeating-radial-gradient` 的渐变与非重复版本具有相同语法。

受 Firefox 3.6、Chrome 10 和 WebKit 每日构建版本（因此还有 Safari 6）支持，下面是使用几行 CSS 可创建的效果示例（为了简单起见，这里再次仅使用了带

-webkit-前缀的语法):

```
.repeat_linear_1 {
  background-image:
    -webkit-repeating-linear-gradient(left,
      rgba(0,0,0,0.5) 10%,
      rgba(0,0,0,0.1) 30%);
}
.repeat_radial_2 {
  background-image:
    -webkit-repeating-radial-gradient(top left, circle,
      rgba(0,0,0,0.9),
      rgba(0,0,0,0.1) 10%,
      rgba(0,0,0,0.5) 20%);
}
.multiple_gradients_3 {
  background-image:
    -webkit-repeating-linear-gradient(left,
      rgba(0,0,0,0.5) 10%,
      rgba(0,0,0,0.1) 30%),
    -webkit-repeating-radial-gradient(top left, circle,
      rgba(0,0,0,0.9),
      rgba(0,0,0,0.1) 10%,
      rgba(0,0,0,0.5) 20%);
}
```

生成的渐变如图 7.13 所示。



图 7.13 一些重复渐变示例

7.4 多背景图像

您可能已注意到，我们具有线性渐变的广告是不完整的：我们缺少了自行车。在 CSS3 之前，添加自行车将需要在标记中放置一个额外的元素以包含新背景图像。在 CSS3 中，无需为每个背景图像包含一个元素，它为我们提供了向任何元素（甚

至伪元素)添加多个背景图像的能力。

要理解多背景图像,需要理解各种背景属性的语法和值。无论拥有一个还是多个背景图像,所有背景属性(包括 `background-image` 和简写的 `background` 属性)的值的语法都是相同的。要声明多背景图像,只需将每个图像的值用逗号分隔。例如:

```
background-image:
  url(firstImage.jpg),
  url(secondImage.gif),
  url(thirdImage.png);
```

也可以使用简写的 `background` 属性:

```
background:
  url(firstImage.jpg) no-repeat 0 0,
  url(secondImage.gif) no-repeat 100% 0,
  url(thirdImage.png) no-repeat 50% 0;
```

背景图像彼此重叠在一起,第一个声明在最顶部,就好像它拥有较高的 `z-index` 一样。最后一个图像位于声明中在它之前的所有图像之下,就好像它拥有较低的 `z-index` 一样。基本而言,可将图像视为按反方向堆叠:第一个在最顶部,最后一个在最底部。

如果希望声明一种背景颜色,尤其是它的浅色文本位于深色背景图像上时,应该最后声明它。使用 `background-color` 属性单独声明它,这样常常更简单且更具有可读性。

作为提醒,简写的 `background` 属性是 8 个普通写法的背景属性的简写。如果使用简写法,声明中省略的普通写法的背景属性值将默认设置为普通写法属性的默认(或初始)值。各种背景属性的默认值如下所示。

```
■ background-color: transparent;
■ background-image: none;
■ background-position: 0 0;
■ background-size: auto;
■ background-repeat: repeat;
```



```

■ background-clip: border-box;

■ background-origin: padding-box;

■ background-attachment: scroll;

```

就像单个背景图像的声明一样，可以包含一个渐变作为多个背景图像之一。我们的广告中就是这么做的。为简短起见，仅显示了不带前缀的版本。以类似的方式将自行车图像包含在每个 `background-image` 声明中：

css/styles.css (excerpt)

```

#ad2 {
  :
  background-image:
    url(../images/bg-bike.png),
    linear-gradient(top,
      rgba(0,0,0,0.4) 0,
      rgba(0,0,0,0) 37%,
      rgba(0,0,0,0) 83%,
      rgba(0,0,0,0.06) 92%,
      rgba(0,0,0,0) 98%);
  background-position: 50% 88%, 0 0;
}

```

请注意，我们将自行车图片作为一系列背景图像的第一个，因为我们希望自行车位于渐变之上，而不是位于它之下。我们声明了每个图像的背景位置，以在 `background-image` 属性中声明图像的相同顺序放置它们。如果仅声明了一组值，例如 `background-position: 50% 88%;`，那么所有图像将拥有相同的背景位置，就好像声明了 `background-position: 50% 88%, 50% 88%;`。在这种情况下，`50% 88%` 定位自行车，它是第一个声明的，而 `0 0`（或 `left top`）定位渐变。

因为浏览器将仅关注一个 `background-image` 属性声明（无论它声明了一个还是多个图像），所以自行车图像必须包含在每个 `background-image` 声明中，因为它们将针对不同的浏览器。请记住，浏览器会忽略它们无法理解的 CSS。所以，如果 Safari 无法理解 `-moz-linear-gradient`（它确实无法理解），它将忽略整个属性/值对。

我们的注册表单的顶部有两个背景图像。在此情况下尽管可以附加一个更宽的图像来覆盖整个表单，但没有这个必要！使用多个背景图像，CSS3 支持附加两个独立的小图像，或者将一个图像画面复制到两个不同的背景位置。这不仅节省了带宽，还在页面顶部需要拉伸时带来了好处：一个图像无法适应具有不同大小的元素。

这一次，我们将使用背景简写：

```
background:
  url(../images/bg-formtitle-left.png) left 13px no-repeat,
  url(../images/bg-formtitle-right.png) right 13px no-repeat;
```



背景简写

当所有可用的背景属性都受支持时，以下两条语句将等效：

```
div {
  background: url("tile.png") no-repeat scroll center
  bottom / cover rgba(0, 0, 0, 0.2);
}
```

```
div {
  background-color: rgba(0,0,0,0.2);
  background-position: 50% 100%;
  background-size: cover;
  background-repeat: no-repeat;
  background-clip: border-box;
  background-origin: padding-box;
  background-attachment: scroll;
  background-image: url(form.png);
}
```

但是，目前由于只有部分浏览器支持所有可用的值，因此我们建议在简写声明中包含 color、position、repeat、attachment 和 image，后面跟上 clip、origin 和 size，或者完全避免简写。必须在普通写法属性之前声明简写，因为任何未在简写中显式声明的值将被视为您声明了默认值。

7.5 背景大小

background-size 属性可用于指定希望背景图像拥有的大小。在理论上，可以在简写的 background 声明中包含 background-size，将它添加到背景位置之后并以斜杠 (/) 分隔。事实证明，没有浏览器能理解这种简写；事实上这将导致它们忽略整个 background 声明，因为它们将它视为不正确的格式。因此，需要将 background-size 属性用作一个独立的声明。

支持 background-size 的浏览器如下所示。

- ❏ Opera 11.01 及更高版本: `background-size` (不带前缀)。
- ❏ Safari 和 Chrome: 最新版本支持不带前缀, 但老版本需要 `-webkit-background-size`。
- ❏ Firefox: `-moz-background-size` 用于 3.6 版本, `background-size` 用于 4 及更高版本。
- ❏ IE9: `background-size`。

可以看到, 采用此语法的不带前缀版本非常快, 它是一个简单的属性, 具有不太可能更改的直观实现。这个示例很好地解释了为什么应该在 CSS 中始终包含不带前缀的版本。

如果以像素形式声明背景图像大小, 就请小心地避免图像扭曲, 要么定义宽度, 要么定义高度, 而不要二者都定义, 将另一个值设置为 `auto`。这将保持图像的长宽比。如果仅包含一个值, 第二个值就会假设为 `auto`。换句话说, 以下两行具有相同的含义:

```
background-size: 100px auto, auto auto;
background-size: 100px, auto auto;
```

与所有背景属性一样, 使用逗号分隔所声明的每个图像的值。如果我们希望自行车显示得大些, 可以这样声明:

```
-webkit-background-size: 100px, cover;
-moz-background-size: 100px, cover;
-o-background-size: 100px, cover;
background-size: 100px auto, cover;
```

通过仅声明图像的宽度, 第二个值将默认为 `auto`, 浏览器将基于长宽比来确定图像的正确高度。

背景图像的默认大小为图像的实际大小。有时图像比它的容器稍微大或小一些。可以以像素(如上所示)或百分比形式定义背景图像的大小, 或者可以使用 `contain` 或 `cover` 关键字。

`contain` 值在缩放图像的同时保持其长宽比, 这可能留下未覆盖的空间。`cover` 值缩放图像, 使它完全覆盖元素。如果元素和它的背景图像具有不同的长宽比, 这可能导致图像被裁剪。



屏幕像素密度或 DPI

`background-size` 属性为具有不同像素密度的设备（比如最新一代的智能电话）带来了很大方便。例如，iPhone 4 的像素密度是以前版本的 4 倍，但为了避免为较旧电话设计的页面看起来太小，iPhone 4 上的浏览器的行为就像它仅拥有 320 像素 × 480 像素的显示屏一样。在本质上，CSS 中的每个像素都对应 4 个屏幕像素。图像会按比例放大来适应屏幕，但这意味着与所显示的文本的平滑度相比，它们有时可能看起来比较粗糙。

要解决此问题，可以向 iPhone 4 提供更高分辨率的图像。例如，为 iPhone 提供一个高分辨率的自行车图像，它的大小为 74 像素 × 90 像素，而不是 37 像素 × 45 像素。但是，我们实际上并不希望图像变为原来的两倍！我们仅希望它占据 37 像素 × 45 像素的空间。可以使用 `background-size` 确保我们的高分辨率图像将占据正确的空间量：

```
-webkit-background-size: 37px 45px, cover;
-moz-background-size: 37px 45px, cover;
-o-background-size: 37px 45px, cover;
background-size: 37px 45px, cover;
```

7.6 小结

CSS3 背景和渐变就介绍到这里。下一章将介绍转换、动画和过渡。这些功能可用于向页面添加动态效果和动作，而不依靠消耗大量带宽和处理器资源的 JavaScript。

CSS3 转换和过渡

我们的页面是静态的。实际上，它是完全静态的。在第 4 章，我们简单学习了如何基于表单的状态，使用`:invalid`和`:valid`伪类调整它的外观。但如何实际移动一些元素呢？如何更改元素的外观——旋转或倾斜它们呢？

多年来，Web 设计人员依靠 JavaScript 实现页面内的动画，并且以一定角度显示文本的唯一方式是使用图像。这种方法不太理想。进入 CSS3 时代，无需任何 JavaScript 或 JPEG，可以轻松倾斜、缩放、移动以及翻转元素。

下面看一下如何完成这些任务。

8.1 转换

Firefox 3.5 及更高版本、Opera 10.5、WebKit 3.2 (Chrome 1) 及更高版本以及甚至 Internet Explorer 9 都支持 CSS3 `transform` 属性，该属性可用于平移、旋转、缩放或倾斜页面上的任何元素。虽然其中一些效果可以使用已经存在的 CSS 功能（比如相对和绝对定位）实现，但是 CSS3 带来了对元素外观的更多方面的前所未有的控制力度。

我们使用**转换函数**操作元素的外观。`transform` 属性的值是一个或多个转换函数（使用空格分隔），它们将按提供的顺序应用。在本书中，我们将介绍所有二维转换函数。WebKit 还支持 3D 空间的元素转换（3D 转换），但这不属于本书的介绍范围。

为了演示转换的工作原理，我们将介绍 The HTML5 Herald 中的另一个广告块，如图 8.1 所示。



图 8.1 此广告块将用于演示 CSS3 转换

8.1.1 平移

平移函数可用于向上、下、左或右移动元素。这些函数类似于 `position: relative;` 的行为，可在其中声明 `top` 和 `left`。当采用平移函数时，不会影响文档流即可移动元素。

`position: relative` 允许根据元素的当前位置或一个父元素或其他祖先元素来定位元素。与它不同，平移的元素仅可相对于其当前的位置移动。

`translate(x,y)` 函数沿 `x` 轴从左往右移动，沿 `y` 轴从上往下移动：

```
-webkit-transform: translate(45px, -45px);
-moz-transform: translate(45px, -45px);
-ms-transform: translate(45px, -45px);
-o-transform: translate(45px, -45px);
transform: translate(45px, -45px);
```

如果仅希望垂直或水平移动元素，可以使用 `translatex` 或 `translatey` 函数：

```
-webkit-transform: translutex(45px);
-moz-transform: translutex(45px);
-ms-transform: translutex(45px);
-o-transform: translutex(45px);
transform: translutex(45px);

-webkit-transform: translatey(-45px);
-moz-transform: translatey(-45px);
-ms-transform: translatey(-45px);
-o-transform: translatey(-45px);
transform: translatey(-45px);
```

对于我们的广告，假设我们希望在用户将鼠标悬停在单词“dukes”上时，向右移动它，就像被我们彪悍的拳击手击中了一样。在标记中，我们拥有：

```
<h1>Put your <span>dukes</span> up sire</h1>
```

只要鼠标悬停在 h1 上，我们就应用该样式。这将使该效果比仅由将鼠标悬停在跨区元素自身上来触发更加个性：

```
css/styles.css (excerpt)

#ad3 h1:hover span {
  color: #484848;
  -webkit-transform: translateX(40px);
  -moz-transform: translateX(40px);
  -ms-transform: translateX(40px);
  -o-transform: translateX(40px);
  transform: translateX(40px);
}
```

这适用于大部分浏览器，但您可能已注意到，其中不包括 WebKit。原因何在？事实证明，WebKit 将仅允许转换块级元素，不支持内联元素。这个问题很容易修复，我们只需向跨区元素中添加 `display: inline-block;`：

```
css/styles.css (excerpt)

#ad3 h1 span {
  font-size: 30px;
  color: #999999;
  display: inline-block;
  ...
}
```

结果如图 8.2 所示。

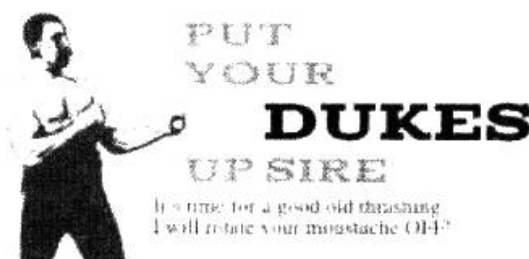


图 8.2 我们的 translate 转换的结果

效果很好，但我们还可以做得更好。让我们看一下如何放大文本，使它变得更大。

8.1.2 缩放

`scale(x,y)` 函数按定义的水平 and 垂直系数来缩放元素。如果仅提供了一个值，它将同时用于 `x` 和 `y` 缩放。例如，`scale(1)` 将保持元素相同大小，`scale(2)` 会将元素长宽放大一倍，`scale(0.5)` 会将元素的长宽减半，等等。也许您已想到，提供不同的值将扭曲元素：

```
-webkit-transform: scale(1.5,0.25);
-moz-transform: scale(1.5,0.25);
-ms-transform: scale(1.5,0.25);
-o-transform: scale(1.5,0.25);
transform: scale(1.5,0.25);
```

与 `translate` 一样，也可以使用 `scalex(x)` 或 `scaley(y)` 函数。这些函数将仅缩放水平尺寸或垂直尺寸。它们分别等效于 `scale(x,1)` 和 `scale(1,y)`。

缩放的元素将远离其中心向外增长或朝其中心向内缩小；换句话说，在尺寸更改时，元素的中心将保持在相同位置。要更改此默认行为，可以包含 `transform-origin` 属性，我们稍后将介绍它。

我们向跨区元素添加一个 `scale` 转换：

css/styles.css (excerpt)

```
#ad3 h1:hover span {
  color: #484848;
  -webkit-transform: translateX(40px) scale(1.5);
  -moz-transform: translateX(40px) scale(1.5);
  -ms-transform: translateX(40px) scale(1.5);
  -o-transform: translateX(40px) scale(1.5);
  transform: translateX(40px) scale(1.5);
}
```

请注意，无需声明新转换，可以使用以空格分隔的转换函数列表提供它，所以我们只需将 `scale` 添加到列表末尾。

值得记住的是，与平移一样，缩放不会影响到文档流。这意味着，如果缩放内部文本，它周围的文本不会重新排版来适应它。图 8.3 中的示例表明这可能会出现。在遇到这种情况时，您可能希望考虑简单地调整元素高度、宽度或字号，而不使用缩放转换。更改这些属性将更改浏览器分配给元素的空间。

Transforming inline text

图 8.3 在嵌入式文本上使用 scale 函数可能导致不想要的结果

但是，在我们的示例中，我们希望文本从广告中突显出来，而不重新排版周围的元素，所以缩放能够执行我们希望它执行的操作。图 8.4 显示了向现有平移添加缩放之后，广告在悬停状态的样子。



图 8.4 我们的广告现在拥有出色的突出效果

这看起来很不错，但仍然可以添加更多效果。

8.1.3 旋转

`rotate()` 函数围绕原点（与 `scale` 一样，默认情况下这是元素的中心）将元素旋转指定的角度。一般而言，角度按度数来声明，正度数会顺时针旋转，负度数会逆时针旋转。除了度数，也可以以百分比、弧度或圈数来提供值，但我们会坚持使用度数。

让我们向“dukes”添加 `rotate` 转换：

```
#ad3 h1:hover span {
  color: #484848;
  -webkit-transform: rotate(10deg) translateX(40px) scale(1.5);
  -moz-transform: rotate(10deg) translateX(40px) scale(1.5);
  -ms-transform: rotate(10deg) translateX(40px) scale(1.5);
  -o-transform: rotate(10deg) translateX(40px) scale(1.5);
  transform: rotate(10deg) translateX(40px) scale(1.5);
}
```

我们沿顺时针方向将 `span` 旋转 10° ，向文本效果中添加被强有力的上钩拳击中的效果。我们在 `translate` 之前声明旋转，因为它会首先应用——请记住转换按提供的顺序应用。有时这将没有任何作用，如果与您想要的效果不同，则需要调整转换的顺序。

最终的转换文本如图 8.5 所示。



图 8.5 文本已平移、放大和旋转——这是一记猛拳

我们还要介绍一种类型的转换。它不会被用于 The HTML5 Herald 中，不过我们也可以看看它。

8.1.4 倾斜

`skew(x,y)` 函数指定沿 X 轴和 Y 轴的一种倾斜。正如您所期望的，`x` 指定 X 轴上的倾斜，`y` 指定 Y 轴上的倾斜。如果省略了第二个参数，`skew` 将仅在 X 轴上发生倾斜：

```
-webkit-transform: skew(15deg, 4deg);
-moz-transform: skew(15deg, 4deg);
-ms-transform: skew(15deg, 4deg);
-o-transform: skew(15deg, 4deg);
transform: skew(15deg, 4deg);
```

例如，将上面的样式应用到标题，会生成如图 8.6 所示的倾斜效果。

A Skewed Perspective

与 `translate` 和 `scale` 一样，倾斜转换也有特定于某个轴的版本：`skewx()` 和 `skewy()`。

图 8.6 一些应用了倾斜转换的文本

8.1.5 更改转换的原点

前面已经暗示过，可以控制应用转换的原点。这使用 `transform-origin` 属性来完成。它具有与 `background-position` 属性相同的语法，默认值为对象的中心（所以缩放和旋转默认将围绕方框的中心进行）。

我们假设转换一个圆。因为默认的 `transform-origin` 为圆的中心，将 `rotate` 转换应用到圆上将没有可见的效果——旋转 90° 的圆看起来与在旋转之前完全相同。但是，如果将圆的 `transform-origin` 设置为 `10% 10%`，您将可以看到圆的旋转，如图 8.7 所示。

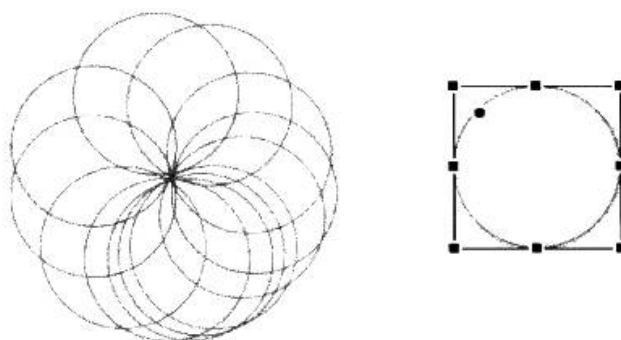


图 8.7 旋转圆仅在设置了 transform-origin 时有效

在 WebKit、Firefox 和 Opera 中，需要在 transform-origin 属性上添加供应商前缀：

```
-webkit-transform-origin: 0 0;
-moz-transform-origin: 0 0;
-o-transform-origin: 0 0;
transform-origin: 0 0;
```

8.1.6 对 Internet Explorer 8 及更早版本的支持

虽然 Internet Explorer 6、Internet Explorer 7 或 Internet Explorer 8 中不支持 CSS3 转换，但是可以使用其他 CSS 属性（包括滤镜）模拟这些效果。要“平移”，可以使用 position:relative;，以及 top 和 left 值：

```
.translate {
  position: relative;
  top: 200px;
  left: 200px;
}
```

也可通过更改元素的宽度和高度来缩放它。但是，请记住，转换的元素仍然会占用它们在缩放之前占用的空间，更改宽度或高度会更改分配给元素的空间，可能会影响布局。

甚至可以使用滤镜在 Internet Explorer 中旋转元素，但效果比较难看：

```
.rotate {
  transform: rotate(15deg);
  filter: progid:DXImageTransform.Microsoft.Matrix(
    sizingMethod='auto expand', M11=0.9659258262890683,
    M12=-0.25881904510252074, M21=0.25881904510252074,
```



```

        M22=0.9659258262890683);
-ms-filter: "progid:DXImageTransform.Microsoft.Matrix(
    M11=0.9659258262890683, M12=-0.25881904510252074,
    M21=0.25881904510252074, M22=0.9659258262890683,
    sizingMethod='auto expand')";
zoom: 1;
}

```

此滤镜的语法在这里没有必要深入探讨。如果希望在 Internet Explorer 中旋转元素, 请访问 <http://css3please.com/>, 获得指定旋转效果的跨浏览器代码。只要编辑任何 `rotation` 值, 其他版本就会相应地更新。

8.2 过渡

虽然在 Internet Explorer 9 中实现一项功能非常有趣, 但是现在是时候撇开该浏览器了。尽管 Opera、Firefox 和 WebKit 都支持 CSS 过渡, 但 Internet Explorer 仍然是个例外。

过渡允许 CSS 属性的值不断更改, 提供简单的动画效果。例如, 如果链接在鼠标悬停时更改颜色, 可以让它平缓地从一种颜色淡出到另一种, 而不是突然更改。

类似地, 可以为我们刚刚看到的任何转换设置动画, 使页面看起来更加动态。

动画在 JavaScript 中无疑已存在很长时间, 但原生 CSS 过渡在客户端上需要的处理资源少得多, 所以它们通常会显得更加平滑。尤其是在具有有限计算功率的移动设备上, 这能够省去很多麻烦。

CSS 过渡与元素上的常规样式一起声明。只要目标属性更改, 浏览器就会应用过渡。最常见的情况是, 向悬停效果应用不同的样式而引起属性更改。但是, 如果关注的属性是使用 JavaScript 更改的, 那么过渡将具有相同的效果。这很重要: 无需在 JavaScript 中编写动画, 只需更改一个属性值并依靠浏览器来执行所有重要工作。

以下是仅使用 CSS 创建简单过渡的步骤。

- (1) 在默认样式声明中声明元素的初始状态。
- (2) 声明过渡元素的最终状态, 比如悬停状态。
- (3) 在默认样式声明中包含过渡函数, 使用一些不同的属性: `transition-`

`property`、`transition-duration`、`transition-timing-function` 和 `transition-delay`。我们稍后将介绍每个属性及其工作原理。

需要注意的重要一点是，过渡是在默认状态中声明的。目前，过渡函数需要包含供应商前缀 `-webkit-`、`-o-` 和 `-moz-`，分别针对 WebKit、Opera 和 Firefox。

需要理解的内容可能很多，所以让我们逐个介绍一下各种过渡值。在介绍的过程中，我们将向上一节中添加到广告中的转换应用一种过渡，使单词“**dukes**”在鼠标悬停时平滑地移动到其新位置。

8.2.1 transition-property

`transition-property` 列出元素应该过渡的 CSS 属性。可过渡的属性包括背景、边框和方框模型属性。可以过渡字号和字体粗细，但字体集不能过渡。W3C 于 2010 年 8 月最后更新了可过渡的属性列表：

- `background-color` 和 `background-position`;
- `border-color`、`border-spacing` 和 `border-width`;
- `bottom`、`top`、`left` 和 `right`;
- `clip`;
- `color`;
- `crop`;
- `font-size` 和 `font-weight`;
- `height` 和 `width`;
- `letter-spacing`;
- `line-height`;
- `margin`;
- `max-height`、`max-width`、`min-height` 和 `min-width`;
- `opacity`;
- `outline-color`、`outline-offset` 和 `outline-width`;

```
padding;
text-indent;
text-shadow;
vertical-align;
visibility;
word-spacing;
z-index。
```

在一些浏览器中有更多属性可以过渡，包括转换函数，但它们还未包含在建议规范中。另请注意，在编写本书时，不是所有浏览器都支持所有上述属性的过渡。

可以向 `transition-property` 声明提供任意数量的 CSS 属性，以逗号分隔。也可以使用关键字 `all` 来表明每个支持的属性都应该应用动画。

在我们的广告中，我们将向 `transform` 属性应用过渡：

```
#ad2 h1 span {
  -webkit-transition-property: -webkit-transform;
  -moz-transition-property: -moz-transform;
  -o-transition-property: -o-transform;
  transition-property: transform;
}
```

请注意，我们需要指定属性的前缀形式——例如，无法在仅能理解 `-moz-transform` 的浏览器中为 `transform` 应用动画。

因为可过渡的属性列表在不断改变，所以请留意包含的属性：可能一个属性在编写页面时无法应用动画，但它最终将能够应用，所以在指定属性时请精心选择，只有在真正希望为每个属性应用动画时才使用 `all`。

目前为止，这些样式没有任何效果，这是因为我们还需要指定过渡的持续时间。

8.2.2 transition-duration

`transition-duration` 属性设置过渡将持续的时间。可以以秒 (s) 或毫秒 (ms) 为单位指定此值。我们希望动画非常快，所以将指定 0.2 秒或 200 毫秒：


```
-webkit-transition-duration: 0.2s;
-moz-transition-duration: 0.2s;
-o-transition-duration: 0.2s;
transition-duration: 0.2s;
```

有了这些样式，我们的 `span` 将在鼠标悬停时过渡。请注意，“反向”过渡（元素返回到其以前的位置）也将在相同持续时间内发生。



自动平缓地降级

尽管只有部分浏览器支持过渡，但它们与会更改这些变更含义的属性独立地声明这一事实在不支持过渡的浏览器中仍然很明显。这些浏览器仍然能够很好地应用 `:hover`（或其他）状态，除非更改将立即发生，而不是经历过渡。

8.2.3 transition-timing-function

`transition-timing-function` 可用于更细粒度地控制过渡的速度。希望过渡缓慢开始并逐渐加快、快速开始并缓慢结束、匀速进行或以其他某种进度进行？可以指定关键字 `ease`、`linear`、`ease-in`、`ease-out` 或 `ease-in-out` 之一。熟悉它们的最佳方式是试用每一个。在大部分情况下，人们将对您想要创建的效果感到满意。请记住在测试计时功能时设定相对较长的 `transition-duration`——如果太快，将无法看出差别。

除了这 5 个关键字，也可以使用 `cubic-bezier()` 函数更准确地描述您的计时函数。它接受 4 个数字参数，例如，`linear` 相当于 `cubic-bezier(0.0, 0.0, 1.0, 1.0)`。如果您学了 6 年的微积分，那么编写 3 次 Bézier 函数的方法可能很有用；否则，您可能希望坚持使用 5 个基本的计时函数。也可以利用允许处理不同值的在线工具，比如 <http://www.netzgesta.de/dev/cubic-bezier-timing-function.html>。

对于我们的过渡，我们将使用 `ease-out`：

```
-webkit-transition-timing-function: ease-out;
-moz-transition-timing-function: ease-out;
-o-transition-timing-function: ease-out;
transition-timing-function: ease-out;
```

这使过渡迅速开始，然后逐渐变慢。当然，在 0.2 秒的持续时间内，几乎察觉

不到区别。

8.2.4 transition-delay

最后, 通过使用 `transition-delay` 属性, 也可以在动画开始之前引入延迟。通常, 过渡会迅速开始, 所以默认值为 0。包含毫秒 (ms) 或秒 (s) 数以延迟过渡:

```
-webkit-transition-delay: 250ms;
-moz-transition-delay: 250ms;
-o-transition-delay: 250ms;
transition-delay: 250ms;
```



负延迟

有趣的是, 比整个动画持续时间更短的负时间延迟将导致它迅速开始, 但它将从动画的中途开始。例如, 在 2 秒的动画上具有延迟 -500 毫秒, 过渡将从动画的 1/4 处开始, 持续 1.5 秒。这可以用于创建一些有趣的效果, 所以值得注意。

8.2.5 transition 简写属性

使用 4 个过渡属性和 3 个供应商前缀, 一个过渡最终可能需要 16 行 CSS 代码。幸运的是, 与其他属性一样, 可以使用简写法。`transition` 属性是上述 4 个过渡函数的简写形式。我们再看一下目前为止我们的过渡:

```
#ad2 h1 span {
  -webkit-transition-property: -webkit-transform, color;
  -moz-transition-property: -moz-transform, color;
  -o-transition-property: -o-transform, color;
  transition-property: transform, color;
  -webkit-transition-duration: 0.2s;
  -moz-transition-duration: 0.2s;
  -o-transition-duration: 0.2s;
  transition-duration: 0.2s;
  -webkit-transition-timing-function: ease-out;
  -moz-transition-timing-function: ease-out;
  -o-transition-timing-function: ease-out;
  transition-timing-function: ease-out;
}
```

现在, 将所有这些值结合到一个简写声明中:

css/styles.css (excerpt)

```
#ad2 h1 span {
  -webkit-transition: -webkit-transform 0.2s ease-out;
  -moz-transition: -moz-transform 0.2s ease-out;
  -o-transition: -o-transform 0.2s ease-out;
  transition: transform 0.2s ease-out;
}
```

请注意，值的顺序很重要，必须为以下顺序（但无需指定所有 4 个值）。

- (1) transition-property。
- (2) transition-duration。
- (3) transition-function。
- (4) transition-delay。

8.2.6 多个过渡

过渡属性支持在一次调用中使用多个过渡。例如，可以在更改旋转和大小的同时更改颜色。

假设我们不仅要过渡旋转，还要过渡文本的 color 属性。我们必须首先在过渡的样式声明中包含 color 属性，然后在 transition-property 值列表中包含 color 属性，或者使用关键字 all：

```
transition-property: transform, color;
transition-duration: 0.2s;
transition-timing-function: ease-out;
```

也可以为要应用动画的每个属性指定不同的持续时间和计时函数。只需在一个逗号分隔列表中包含每个值（使用 transition-property 中的相同顺序）：

```
transition-property: transform, color;
transition-duration: 0.2s, 0.1s;
transition-timing-function: ease-out, linear;
```

上面的属性将在 0.2 秒内向 transform 应用一个 ease-out 过渡，在 0.1 秒内向 color 应用一个 linear 过渡。

也可以在使用简写的 `transition` 属性时指定多个过渡。在这种情况下，为每个过渡集中指定所有值，使用逗号分隔每个过渡：

```
transition: color 0.2s ease-out, transform 0.2s ease-out;
```

如果希望以相同的速率和延迟更改两个属性，可以包含两个属性的名称或者应用 `all` 关键字，因为将应用鼠标悬停状态中列出的所有属性。

当使用 `all` 关键字时，所有属性以相同的速率、速度和延迟过渡：

```
-webkit-transition: all 0.2s ease-out;
-moz-transition: all 0.2s ease-out;
-o-transition: all 0.2s ease-out;
transition: all 0.2s ease-out;
```

如果不希望所有属性以相同速率过渡，或者希望选定的一些属性具有过渡效果，可以以逗号分隔列表形式包含各种过渡属性，至少包含每个属性的 `transition-property` 和 `transition-duration`。

8.3 动画

过渡在一定时间内向元素应用动画，但它们的功能有限。可以定义开始状态和结束状态，但不能对任何中间状态进行细粒度的控制。与过渡不同，CSS 动画可通过关键帧控制动画的每一步。如果以前使用过 Flash，可能非常熟悉关键帧的概念；如果未使用过也不要担心，它非常简单。关键帧是一个快照，定义任何平滑过渡的起点或终点。使用 CSS 过渡，我们基本上只能定义第一个和最后一个关键帧。CSS 动画可用于添加任意数量的中间关键帧，以更加复杂的方式引导动画。

在编写本书时，只有 WebKit 支持 CSS 动画。这意味着桌面上的支持很有限，但移动设备上的支持还不错，因为 iOS 和 Android 上的默认浏览器都在 WebKit 上运行。我们前面已经提到，许多移动设备上缺乏强大的处理器，这使 CSS 动画成为了复杂的、CPU 密集型的 JavaScript 动画的不错替代品。

8.3.1 关键帧

要在 CSS 中为元素应用动画，首先要创建一个已命名的动画，然后将它附加到该元素属性声明块中的一个元素上。动画本身不执行任何操作；为了向元素应用动

画，需要将动画与元素相关联。

要创建动画，可使用@keyframes 规则（或者对于当前的 WebKit 实现，使用 @-webkit-keyframes），后跟所选择的名称，该名称用于动画的标识符。然后可以指定关键帧。

对于名为 myAnimation 的动画，@keyframes 规则可能类似于：

```
@-webkit-keyframes 'myAnimation'{
  /* put animation keyframes here */
}
```

每个关键帧看起来就像它自己嵌套的 CSS 声明块。但是，无需使用选择器，可以使用关键字 from 或 to、百分比值，或者一个以逗号分隔的百分比值列表。此值指定关键帧位于动画中的位置。

在每个关键帧内，包含目标属性和值。在每个关键帧之间，浏览器的动画引擎将平滑地插入值。

关键帧可以任何顺序指定，动画中的关键帧顺序由百分比值而不是声明的顺序确定。

以下是一些简单的动画：

```
@-webkit-keyframes 'appear' {
  0% {
    opacity: 0;
  }
  100% {
    opacity: 1;
  }
}

@-webkit-keyframes 'disappear' {
  to {
    opacity: 0;
  }
  from {
    opacity: 1;
  }
}

@-webkit-keyframes 'appearDisappear' {
  0%, 100% {
    opacity: 0;
  }
}
```

```

    20%, 80% {
        opacity: 1;
    }
}

```

最后一个动画值得额外注意：我们向 0% 与 100% 以及 20% 与 80% 应用了相同的样式。在这种情况下，它意味着元素最初是不可见的（`opacity: 0;`），渐渐淡入，在持续时间的 20% 处变为可见，保持可见直到 80%，然后淡出。

我们创建了 3 个动画，但它们未附加到任何元素上。定义动画后，下一步是使用各种动画属性将它应用到一个或多个元素。

8.3.2 动画属性

WebKit 支持的动画属性如下所示，包含 `-webkit-` 供应商前缀。

1. animation-name

此属性用于将动画（使用前面的 `@keyframes` 语法）附加到元素上：

```
-webkit-animation-name: 'appear';
```

请注意，属性值和 `@keyframes` 选择器中对动画名称的引用是可选的。我们建议包含它们，以保持样式尽可能易读，并且避免冲突。

2. animation-duration

`animation-duration` 属性定义动画完成一次迭代（从 0% 到 100%）所花的时间长度（以秒或毫秒为单位）：

```
-webkit-animation-duration: 300ms;
```

3. animation-timing-function

类似于 `transition-timing-function` 属性，`animation-timing-function` 确定动画在其持续时间内的进度。它的选项与 `transition-timing-function` 相同，即 `ease`、`linear`、`ease-in`、`ease-out`、`ease-in-out` 或 `cubic-bezier`：

```
-webkit-animation-timing-function: linear;
```


4. animation-iteration-count

此属性用于定义动画将播放多少次。该值通常为整数，但也可以使用带有小数点的数字（在这种情况下，动画将在一次迭代的中途结束，或者设置值 `infinite` 以无限地重复动画。如果省略，它将默认为 1，在这种情况下，动画将仅发生一次：

```
-webkit-animation-iteration-count: infinite;
```

5. animation-direction

当动画迭代时，可以使用具有值 `alternate` 的 `animation-direction` 属性来使其他每次迭代反向播放动画。例如，在弹跳球动画中，可以为落下的球提供关键帧，然后使用 `animation-direction: alternate;` 来在播放动画的每秒中反转它：

```
-webkit-animation-direction: alternate;
```

默认值为 `normal`，所以动画将在每次迭代中正向播放。

当动画反向播放时，计时函数也会反转，例如，`ease-in` 会变为 `ease-out`。

6. animation-delay

用于定义在浏览器开始执行动画之前等待的毫秒数或秒数：

```
-webkit-animation-delay: 15s;
```

7. animation-fill-mode

`animation-fill-mode` 属性定义在动画开始之前和结束之后发生的操作。默认情况下，动画不会影响它的迭代之外的属性值，但使用 `animation-fillmode`，可以改写此默认行为。我们告诉动画在第一个关键帧上等待动画开始，或者在动画结束时停在最后一个关键帧上而不反转到初始值，或者同时定义二者。

可用的值包括 `none`、`forwards`、`backwards` 或 `both`。默认值为 `none`，在这种情况下动画将按预期进行和结束，在动画完成其最终迭代时反转到初始关键帧。当设置为 `forwards` 时，动画在结束后继续应用最后的关键帧的值。当设置为 `backwards` 时，会在向元素应用动画样式时迅速应用动画的初始关键帧。如您所

想, both 同时应用 backwards 和 forwards 效果:

```
-webkit-animation-fill-mode: forwards;
```

8. animation-play-state

animation-play-state 属性定义动画运行还是暂停。暂停的动画静态显示动画的当前状态。当暂停的动画恢复时, 它从当前位置重新开始运行。这提供了一种简单方式来从 JavaScript 控制 CSS 动画。

9. 简写的 animation 属性

幸运的是, 所有这些动画属性有一种简写法。animation 属性以一个以空格分隔的值列表形式接受普通写法的 animation-name、animation-duration、animation-timing-function、animation-delay、animation-iteration-count、animation-direction 和 animation-fill-mode 属性的值:

```
.verbose {
  -webkit-animation-name: 'appear';
  -webkit-animation-duration: 300ms;
  -webkit-animation-timing-function: ease-in;
  -webkit-animation-iteration-count: 1;
  -webkit-animation-direction: alternate;
  -webkit-animation-delay: 5s;
  -webkit-animation-fill-mode: backwards;
}

/* shorthand */
.concise {
  -webkit-animation: 'appear' 300ms ease-in 1 alternate 5s
  ➡ backwards;
}
```

要在元素上声明多个动画, 可以包括每个动画名称的分组, 每个简写的分组以逗号分隔。例如:

```
.target {
  -webkit-animation:
    'animationOne' 300ms ease-in 0s backwards,
    'animationTwo' 600ms ease-out 1s forwards;
}
```

8.4 小结

使用转换、过渡和动画，我们的网站看起来更加动态。但请记住一句古老的格言：只是因为可以做，并不意味着应该去做。动画于 20 世纪 90 年代晚期在网络上大量涌现，我们中许多人都还记得闪动的广告和滚动的选择框，在一定程度上，这些问题如今仍然存在。请在恰当的地方使用动画和过渡，以增强用户体验，而在其他地方忽略它们。

要在 CSS3 中使我们的网站看起来更像旧时的报纸，我们还有许多知识要学。下一章将介绍怎样不依靠 `float` 创建栏，以及如何包含默认未安装在用户计算机上的漂亮字体。

嵌入字体和多列布局

我们向 The HTML5 Herald 添加了许多装饰，但仍然缺少一些关键的要素来使它真正具有旧时的风格。为了看起来像真正的报纸，文章的文本应该分列布局，我们应该使用一些适合该时期的恰当字体。

在本章中，我们将使用 CSS3 列和@font-face 完成网站的最终外观。

9.1 Web 字体和@font-face

在网络诞生之初，设计人员的梦想只是创建具有漂亮版式的网站。但是，众所周知，浏览器仅限于使用用户在其系统上安装的字体呈现文本。实际上，这使大部分网站受限于少量的字体：Arial、Verdana、Times 和 Georgia 等。

多年来，我们已提出了大量针对此问题不错的解决办法。我们创建了 JPEG 和 PNG 来用作网站标题、徽标、按钮和导航元素。当这些元素需要额外的状态或变体时，我们创建了更多图像或图像动画来确保页面保持美观和响应灵敏。只要设计或文本更改，所有这些图像就必须重新创建。

这对页面上一些元素而言可能是一种可接受的解决方案，但要求设计人员在 Photoshop 中设计每篇新文章的标题，然后将它上传到网站，这不太实际。所以，对于需要频繁更改的关键页面元素，我们坚持使用少数字体。

为了填补这一版式上的空白, 诞生了一些基于 Flash 和 JavaScript 的非常优秀的字体嵌入脚本 (比如 sIFR), 以及基于画布的 Cufón。尽管这些方法已是不错的应急措施, 允许我们包含自己的字体, 但它们拥有严重的缺陷。有时它们很难实现, 它们要求启用 JavaScript, 在使用 sIFR 时还要求安装 Flash 插件。此外, 这显著减缓了页面的下载和呈现。

幸运的是, 现在有一种更好的方式: @font-face 是一种针对嵌入字体的纯 CSS 解决方案, 几乎市面上的每种浏览器都支持它, 包括 Internet Explorer 6。

我们将在 The HTML5 Herald 网站上包含两种嵌入字体: 来自 The League of Movable Type 的 League Gothic¹ 和 Ben Weiner of Reading Type 提供的 Acknowledgement Medium²。这两种字体分别如图 9.1 和图 9.2 所示。

League Gothic

图 9.1 League Gothic

ACKNOWLEDGEMENT

图 9.2 Acknowledgement Medium

我们现在看看如何嵌入这些字体, 使用它们来改进网站上的任何文本, 就像它们安装在用户机器上一样。

9.1.1 实现@font-face

@font-face 是多个 CSS@规则之一, 这些规则还包括@media、@import、@page 和我们刚刚看到的@keyframes。@规则是将多项规则封装在一个声明中来充当浏览器 CSS 处理器的指令的方式。@font-face@规则可用于指定自定义字体, 然后可将这些字体包含在其他声明块中。

要使用@font-face 包含字体, 必须:

- (1) 通过各种格式将字体文件加载到服务器上, 以支持所有不同的浏览器;
- (2) 命名、描述并将该字体链接到一个@font-face 规则中;
- (3) 在 font-family 属性值中包含字体的名称, 就像对系统字体所做的一样。

您已知道如何将文件上传到服务器, 所以下一节将探讨各种文件类型的细节。至于现在, 我们将关注第二步和第三步, 让您对@font-face 的语法有所了解。

¹ <http://www.theleagueofmoveabletype.com/>

² <http://www.readingtype.org/>

以下是@font-face 块的一个示例：

```
@font-face {
  font-family: 'fontName';
  src: source;
  font-weight: weight;
  font-style: style;
}
```

字体集和来源是必需的，但字体粗细和样式是可选的。

需要为在网站包含的每种字体包含一个独立的@font-face @规则。还必须为该字体的每种变体（常规、细、粗、斜体和黑体等）包含一个独立的@规则。The HTML5 Herald 将需要两种导入的字体，所以我们将包含两个@font-face 块：

css/styles.css (excerpt)

```
@font-face {
  :
}

@font-face {
  :
}
```

@font-face @规则的 font-family 部分与您熟悉的 font-family 属性稍有不同。在这里，我们声明了字体的名称，而不是向元素分配一种具有指定名称的字体。字体名称可以是您喜欢的任何形式，它仅是字体文件的引用，所以甚至无需与字体的名称相对应。当然，最好使用字体的名称来保持 CSS 可读且可维护。建立一种约定并在所有字体上坚持遵守它，这样做很不错。对于我们的两种字体，我们将使用驼峰式大小写形式：

css/styles.css (excerpt)

```
@font-face {
  font-family: 'LeagueGothic';
}

@font-face {
  font-family: 'AcknowledgementMedium';
}
```


9.1.2 声明字体来源

现在我们已拥有了@font-face 规则的框架布局，为每条规则分配了一个名称，是时候将它们链接到实际的字体文件了。src 属性可接受多种格式。此外，可以声明多个来源。如果浏览器未能找到第一个来源，它将试着查找下一个，依此类推，直到找到一个来源或用尽所有选项。

我们向 League Gothic 声明中添加更多格式：

css/styles.css (excerpt)

```
@font-face {
  font-family: 'LeagueGothicRegular';
  src: url('../fonts/League_Gothic-webfont.eot') format('eot'),
       url('../fonts/League_Gothic-webfont.woff') format('woff'),
       url('../fonts/League_Gothic-webfont.ttf') format('truetype'),
       url('../fonts/League_Gothic-webfont.svg#webfontFHZvtkso')
  ↪format('svg');
}
```

上面的代码块中列出了 4 种字体来源。第一个声明为 EOT 字体声明，这是一种针对 Internet Explorer 的专用格式，是 Internet Explorer 4 至 Internet Explorer 8 唯一能理解的文件类型。

然后我们定义了 WOFF（Web 开放字体格式，一种新兴的标准）、OTF（Open Type）、TTF（TrueType）和 SVG（Scalable Vector Graphics，可缩放矢量图形）字体文件。尽管大部分桌面浏览器都将使用前 3 种声明之一，但还是要确保包含了 SVG 格式，它最初是 iPhone 唯一支持的格式¹。

表 9.1 统计了各种浏览器对不同格式的支持。可以看到，没有哪一种格式受到了所有浏览器的支持，所以我们需要提供多种格式，就像在第 5 章的视频中所做的一样。

表 9.1 字体格式的浏览器支持

| | Internet Explorer | Safari | Chrome | Firefox | Opera | iOS |
|------------|-------------------|-----------|---------|-----------|------------|-----------|
| @font-face | 4 及更高版本 | 3.1 及更高版本 | 4 及更高版本 | 3.5 及更高版本 | 10 及更高版本 | 3.2 及更高版本 |
| WOFF | 9 及更高版本 | 6 及更高版本 | 6 及更高版本 | 3.6 及更高版本 | 11.1 及更高版本 | |

¹ iPhone 最近在 4.2 版中扩展了支持以包含 OTF，但暂时仍然有必要包含 SVG。

续表

| | Internet Explorer | Safari | Chrome | Firefox | Opera | iOS |
|-----|-------------------|-----------|---------|-----------|----------|-----------|
| OTF | | 3.1 及更高版本 | 4 及更高版本 | 3.5 及更高版本 | 10 及更高版本 | 4.2 及更高版本 |
| TTF | 9 及更高版本? | 3.1 及更高版本 | 4 及更高版本 | 3.5 及更高版本 | 10 及更高版本 | 4.2 及更高版本 |
| SVG | | 3.1 及更高版本 | 4 及更高版本 | | 10 及更高版本 | 3.2 及更高版本 |
| EOT | 4 及更高版本 | | | | | |

添加这些额外的字体格式可确保支持每种浏览器，但不幸的是，它将在 Internet Explorer 9 之前的 Internet Explorer 版本中出现问题。这些浏览器将第一个 url 和最后一个之间的所有内容视为一个 URL，所以将无法加载字体。乍看起来，似乎我们必须在支持 Internet Explorer 和支持其他每个浏览器之间进行选择，但幸运的是有一种解决方案。一篇 FontSpring 博客文章¹已详细说明，可以在 EOT URL 末尾添加一个查询字符串。这会诱使浏览器认为 src 属性的剩余部分是查询字符串的延续，所以它将查找正确的 URL 并加载该字体：

(excerpt)

```
@font-face {
  font-family: 'LeagueGothicRegular';
  src: url('../fonts/League_Gothic-webfont.eot?#iefix')
  ↳format('eot'),
      url('../fonts/League_Gothic-webfont.woff') format('woff'),
      url('../fonts/League_Gothic-webfont.ttf') format('truetype'),
      url('../fonts/League_Gothic-webfont.svg#webfontFHvztkso')
  ↳format('svg');
}
```

此语法拥有一个潜在的故障点：Internet Explorer 9 拥有一个称为“兼容性模式”的功能，使用该模式，它将尝试使用与 Internet Explorer 7 或 Internet Explorer 8 相同的方式呈现页面。此模式的引入是为了避免旧网站外观在 Internet Explorer 9 更加符合标准的呈现模式中被破坏。但是，兼容性模式下的 Internet Explorer 9 不会再现加载 EOT 字体时的错误，所以上面的声明将失败。要解决此问题，可以在一个单独的 src 属性中添加一个额外的 EOT URL：

¹ <http://www.fontspring.com/blog/the-new-bulletproof-font-face-syntax>

(excerpt)

```

@font-face {
  font-family: 'LeagueGothicRegular';
  src: url('../fonts/League_Gothic-webfont.eot');
  src: url('../fonts/League_Gothic-webfont.eot?#iefix')
  ➡format('eot'),
    url('../fonts/League_Gothic-webfont.woff') format('woff'),
    url('../fonts/League_Gothic-webfont.ttf') format('truetype'),
    url('../fonts/League_Gothic-webfont.svg#webfontFHzvtkso')
  ➡format('svg');
}

```

这可能是一种不必要的预防措施，因为一般而言，用户在查看会出现此问题的网站时，在将 Internet Explorer 切换到兼容性模式时很慎重。也可以通过在文档的 head 中添加此 meta 元素，强制 Internet Explorer 退出兼容性模式。

```
<meta http-equiv="X-UA-Compatible" content="IE=Edge">
```

也可以添加一个额外的 HTTP 标头来实现相同的结果，这可在 .htaccess 文件（或等效文件）中使用一条指令来完成：

```

<IfModule mod_setenvif.c>
  <IfModule mod_headers.c>
    BrowserMatch MSIE ie
    Header set X-UA-Compatible "IE=Edge"
  </IfModule>
</IfModule>

```

9.1.3 字体属性描述符

也可以添加字体属性描述符（包括 font-style、font-variant 和 font-weight 等）来定义字形的特征，它们也可以用于将样式与特定的字形匹配。这些值与等效的 CSS 属性相同：

```

@font-face {
  font-family: 'LeagueGothicRegular';
  src: url('../fonts/League_Gothic-webfont.eot');
  src: url('../fonts/League_Gothic-webfont.eot?#iefix')
  ➡format('eot'),
    url('../fonts/League_Gothic-webfont.woff') format('woff'),
    url('../fonts/League_Gothic-webfont.ttf') format('truetype'),
    url('../fonts/League_Gothic-webfont.svg#webfontFHzvtkso')

```



```

    ↪format('svg');
    font-weight: bold;
    font-style: normal;
}

```

此行为与您所期望的有所不同。没有告诉浏览器将字体加粗，而是告诉它这是字体的粗体变形。这可能引起混淆，该行为在一些浏览器中也可能比较怪异。

但是，在@font-face 规则声明中使用 font-weight 或 font-style 描述符是有原因的。可以为相同的 font-family 名称声明多个字体来源：

```

@font-face {
  font-family: 'CoolFont';
  font-style: normal;
  src: url(fonts/CoolFontStd.ttf);
}

@font-face {
  font-family: 'CoolFont';
  font-style: italic;
  src: url(fonts/CoolFontItalic.ttf);
}

.whichFont {
  font-family: 'CoolFont';
}

```

请注意，两条@规则都使用了相同的字体集名称，但使用了不同的字体样式。在本例中，.whichFont 元素将使用 CoolFontStd.ttf 字体，因为它匹配在该@规则中指定的样式。但是，如果元素继承一种斜体字体样式，它将切换为使用 CoolFontItalic.ttf 字体。

9.1.4 Unicode 范围

还有一个 unicode-range 描述符，可采用它来定义字体支持的 Unicode 字符的范围。如果省略了此属性，字体文件中包含的所有字符都将可用。

我们不会在网站中使用此描述符，但这里给出了它的一个示例：

```

unicode-range: U+000-49F, U+2000-27FF, U+2900-2BFF, U+1D400-1D7FF;

```

9.1.5 应用字体

使用@font-face 语法声明了字体之后，就可以像 CSS 中的任何普通系统字体一样引用它了：将它包含在一个“堆栈”中，作为 font-family 属性的值。声明一两种回滚字体，以供在嵌入字体失败时加载，这是一种不错的做法。

让我们看一下一个来自 The HTML5 Herald 的示例：

```
css/styles.css (excerpt)

h1 {
  text-shadow: #fff 1px 1px;
  font-family: LeagueGothic, Tahoma, Geneva, sans-serif;
  text-transform: uppercase;
  line-height: 1;
}
```

我们的嵌入字体用在了样式表中多个不同的位置，但为您展示了这一理念。

9.1.6 法律因素

我们在网站上包含了两种字体的标记，但还未放置字体文件本身。我们发现这两种字体都可在线免费获取。它们都以免费软件的形式授权，也就是说，可免费将它们用于个人和商业用途。一般而言，应该将此字体视为应该用于@font-face 的字体，除非使用了第三方服务。

@font-face 与使用图像文件中的某种字体有何区别？在互联网上有了网站之后，字体源文件就会托管在公共的 Web 服务器上，所以从理论上讲，任何人都可下载它们。事实上，为了在页面上呈现文本，浏览器必须下载字体文件。通过使用@font-face，可以将字体分发给访问您网站的任何人。要在网站上包含一种字体，需要获得分发该字体的法律许可。

拥有或购买了字体并不意味着就拥有了重新分发它的法律权利，就像在 iTunes 上购买一首歌曲不会授予您将它上传到网站上供任何人下载一样。与允许在计算机上出于个人或商业用途而使用字体的许可证相比，授权分发字体的许可证更加昂贵（且更少）。

但是，一些网站拥有可通过 Creative Commons¹、共享软件或免费软件许可免费

¹ 如果不熟悉 Creative Commons 许可证，可以在 <http://creativecommons.org/> 上找到更多信息

下载的 Web 字体。也有一些付费和订阅服务可用于购买或租用字体，它们一般提供了现成的脚本或样式表来简化@font-face 的使用。

一些提供 Web 字体服务的网站包括 Typekit¹、Typotheque²、Webtype³、Fontdeck⁴ 和 Fonts.com⁵。

Google 的 Web 字体目录⁶拥有一个不断增大的字体集合，它由 Google 的服务器免费提供和维护。它提供了一个指向包含所有所需@font-face 规则的样式表的 URL，所以需要做的就是将一个 link 元素添加到您的文档，以开始使用一种字体。

当选择服务时，字体选择和价格是重要的考虑因素，但还有其他考虑因素。请确保在选择使用任何服务时考虑了加载速度。字体文件可能非常大，可能包含数千个字符。良好的服务允许选择字符子集，以及字体样式子集，以减小文件大小。另请记住，一些服务需要 JavaScript 才可正常使用。

9.1.7 创建各种字体文件类型：Font Squirrel

如果拥有一种法律允许重新分发的字体，就无需使用任何上述字体服务。但是，必须将您的字体转换为各种必要的格式，以兼容市面上的每种浏览器。那么如何将字体转换为所有这些格式呢？

用于此用途的一个最简单的工具就是 Font Squirrel 的@font-face 生成器⁷。使用此服务，只需几次单击即可从您的桌面选择字体，并将它们转换为 TTF、EOT、WOFF、SVG、SVG 以及 Base64 编码的版本⁸。

默认情况下，选择 Optimal 选项来生成@font-face 工具包，但在一些情况下，可以选择 Expert...并创建一个字符子集来减小文件大小。无需在字体文件中包含每个可想到的字符，可以限制为将在网站上使用的字符。

例如，在 The HTML5 Herald 网站上，Acknowledgement Medium 字体仅用于特定的广告块和标题，所以我们仅需要较小的字符集合。此字体中的所有文本集都是

¹ <http://typekit.com/>

² <http://www.typotheque.com/>

³ <http://www.webtype.com/>

⁴ <http://fontdeck.com/>

⁵ <http://webfonts.fonts.com>

⁶ <http://code.google.com/apis/webfonts/>

⁷ <http://www.fontsquirrel.com/fontface/generator>

⁸ Base64 编码是一种直接将字体文件的所有内容包含在 CSS 文件中的方式。有时这可以避免额外的 HTTP 请求而带来性能优势，但这不属于本书的范围。但不要过分溺爱它，默认设置生成的文件应适用于大部分用途

大写的，所以可以将字体限制为大写字母、标点符号和数字，如图 9.3 所示。



图 9.3 在 Font Squirrel 的@font-face 生成器中选择字符子集

图 9.4 显示了我们从默认字符集中提取的字符子集的文件大小。在我们的示例中，仅包含大写和标点符号的字体比默认字符集小 25%~30%。Font Squirrel 甚至允许为子集指定某些字符，所以如果知道不会使用字母表中的所有字母，则无需包含所有字母。









| | |
|--|-------|
|  acknowledgement-subset.eot | 12 KB |
|  acknowledgement-subset.svg | 25 KB |
|  acknowledgement-subset.ttf | 20 KB |
|  acknowledgement-subset.woff | 12 KB |
|  Acknowledgement-webfont.eot | 29 KB |
|  Acknowledgement-webfont.svg | 37 KB |
|  Acknowledgement-webfont.ttf | 29 KB |
|  Acknowledgement-webfont.woff | 16 KB |

图 9.4 字体子集的文件大小可能小得多

对于 League Gothic 字体，我们将需要一种扩展的字符子集。此字体用于文章标题，它们全部采用大写形式，就像我们的广告一样，所以我们可以省略小写字母，但是应该考虑到标题的内容可能包含更广泛的字符。此外，用户也许会使用浏览器内的工具或 Google 翻译来翻译页面上的内容，这时可能需要其他字符。所以，对于 League Gothic，我们将使用默认的 **Basic Subsetting**，提供西方语言所需的所有字符。

当采用@font-face 时，作为一种一般规则，尽可能合理地最小化字体文件大小，同时确保包含了足够的字符，使网站的翻译版本仍然可用。

上传了字体供处理并选择了所有选项之后，按 **Download Your Kit**。Font Squirrel 提供了一个下载，其中包含所请求的扩展的字体文件、一个针对每种字形样式的演示 HTML 文件，以及一个样式表（可从中直接将代码复制并粘贴到您自己的 CSS 中）。



Font Squirrel 的 Font Catalogue

除了@font-face 生成器，Font Squirrel 网站还包含精选的免费字体的一个目录，它的许可证支持 Web 嵌入。事实上，我们在 The HTML5 Herald 上使用的两种字体都可在 Font Squirrel 上找到，@font-face 字体包已可下载，完全无需依靠生成器。

要适用于所有浏览器，请确保创建了 TTF、WOFF、EOT 和 SVG 字体文件格式。创建了字体文件之后，将 Web 字体上传到您的服务器。复制并粘贴所提供的 CSS，更改路径以指向放置字体的文件夹。确保在@font-face 规则中指定的字体集名称与在样式中使用的规则相匹配，这样基本上就万事俱备了！



排除@font-face 故障

如果字体未能在任何浏览器中显示，问题很可能出在 CSS 中的路径上。请检查以确保字体文件位于它应该在的位置。基于浏览器的调试工具（比如 WebKit 中的 Web Inspector、Opera 中的 Dragonfly 或 Firebug Firefox 扩展）将表明文件是否丢失。

如果确定路径是正确的并且文件位于它应该在的位置，那么请确保服务器已正确配置来提供字体。如果 Windows IIS 服务器无法识别文件的 MIME 类型，则不会提供这些文件，所以请尝试向 MIME 类型列表中添加 WOFF 和 SVG（EOT 和 TTF 应已得到支持）：

```
.woff  application/x-font-woff
.svg   image/svg+xml
```

最后，一些浏览器要求字体从与它们所嵌入的页面所在的域提供。



基于浏览器的开发工具

Safari、Chrome 和 Opera 的标准配置都提供了工具来节省 Web 开发人员的时间。Chrome 和 Opera 已设置了这些工具。只需右键单击（或在 Mac 上按住 Control 键并单击）并选择 **Inspect Element**。将在浏览器底部打开一个面板，突出显示所选元素的 HTML。还会看到应用于该元素的任何 CSS。

尽管 Safari 附带了此工具，但它需要手动启用。要打开它，可以选择 **Safari > Preferences**，然后打开 **Advanced** 选项卡。请确保选中了 **Show Develop menu in menu bar** 复选框。

Firefox 没有附带这样的工具。幸运的是，有一个名为 Firebug 的免费 Firefox 插件提供了相同功能，可以在 <http://getfirebug.com/> 上下载 Firebug。

9.1.8 其他考虑因素

与图像形式的文本相比，嵌入字体可改善性能并缩短维护时间。但请记住，字体文件可能非常大。如果需要为横幅广告使用特定的字体，可以创建一个图像，比包含字体文件更有意义（前提是仅需有限的文本）。

当考虑在网站上包含多个字体文件时，请考虑性能因素。多种字体将增加网站的下载时间，字体的过量使用也可能影响美观。此外，错误的字体可能使内容难以读取。对于正文文本，应该始终坚持使用常用的 Web 安全字体。

值得考虑的另一个因素是，浏览器在完整下载@font-face 字体后才能呈现它。它们在下载完成之前显示内容的行为不尽相同：一些浏览器将使用系统字体呈现文本，而其他浏览器完全不会呈现任何文本。

此效果被 Paul Irish 称为“无样式文本闪烁”或 FOUT¹。要尝试避免此情况发生（或最小化它的持续时间），请保持文件尽可能小，压缩它们，以及在 CSS 文件的标记中尽可能高的位置包含@font-face 规则。如果在源代码中的@font-face 声明之上有一个 script，Internet Explorer 将出现错误，页面在字体下载之后才会呈现内容，所以请确保字体在页面上的脚本上方声明。

另一个减轻@font-face 的性能影响的选项是延迟字体文件的下载，直到页面已呈现。但是，这可能不适用于您的设计人员或客户端，因为它可能导致更明显的 FOUT，即使页面整体加载速度更快²。

当然，我们不希望恐吓您不要使用@font-face，但避免使用这项新获得的自由而不计后果的运行至关重要。请记住，有得必有失，所以请在恰当的地方使用 Web 字体，并考虑可用的备选方案。

9.2 CSS3 多列布局

不用说，“报纸”就像是一行紧凑的文本列。原因如下：新闻将文章分为多列，

¹ <http://paulirish.com/2009/fighting-the-font-face-fout/>

² 关于@font-face 和性能的更多信息，以及如何“延迟加载”字体文件的示例，请访问 <http://www.stevesouders.com/blog/2009/10/13/font-face-and-performance/>

因为太长的文本行不易读取。浏览器窗口可能比印刷的图书更宽，甚至像某些报纸那么宽，所以让 CSS3 为我们提供将内容分为多列的灵活性很有意义。

您可能认为我们始终可以使用 `float` 属性创建列效果，但 `float` 的行为与我们所关注的效果稍有不同。不强制在固定位置分列，报纸样式的列几乎不可能使用 CSS 和 HTML 来实现。确实如此，可以将一篇文章分解为多个 `div`，让每个 `div` 浮动，使其看起来像一组列。但如果内容是动态的怎么办？后端代码将需要确定每列应该在何处开始和结束，以便插入需要的 `div` 标记。

使用 CSS3 列，浏览器确定何时结束一列和开始下一列，而无需任何额外的标记。您保留了更改列数和它们的宽度的灵活性，而无需返回调整页面标记。

现在，最多只能将内容拆分为多列，控制它们的宽度和它们之间的间距。随着支持范围变广，我们将能够分解列，将元素覆盖在多列上，等等。CSS3 列受到的支持比较一般：Firefox 和 WebKit 在多年前就通过带供应商前缀的属性提供了支持，而 Opera 刚刚在 11.10 版中添加了支持（不需要供应商前缀），Internet Explorer 仍然未提供支持。

The HTML5 Herald 主页上几乎所有的内容都分解为列。下面让我们深入分析一下构成 CSS3 列的属性，了解如何在网站上创建这些效果。

9.2.1 column-count 属性

`column-count` 属性指定想要的列数和允许的最大列数。`auto` 的默认值表示元素具有一列。我们最左侧的文章分解为 3 列，广告块下方的文章拥有 2 列：

```
css/styles.css (excerpt)

#primary article .content {
  -webkit-column-count: 3;
  -moz-column-count: 3;
  column-count: 3;
}

#tertiary article .content {
  -webkit-column-count: 2;
  -moz-column-count: 2;
  column-count: 2;
}
```

这就是创建列所需的所有代码。默认情况下，列之间将拥有较小的间距。列的

例如，父元素为 400 像素宽，列间的间距为 10 像素，并且 `column-width` 声明为 150px，那么浏览器将填入两列：

$$(400 \text{ 像素宽} - 10 \text{ 像素列间间距}) \div 150 \text{ 像素宽} = 2.6$$

浏览器四舍五入为两列，使列在所分配的空间内尽可能大，在本例中每列为 195 像素——总宽度减去间距，再除以列数。即使 `column-count` 设置为 3，也仍然只有两列，因为没有足够的空间来包含具有指定宽度的 3 列。换句话说，可以认为 `column-count` 属性指定了最大列数。

列将比 `column-width` 宽度窄的唯一情形是，父元素本身太窄，无法容纳指定宽度的一列。在这种情况下，将有一列填满整个父元素。

以 `em` 为单位声明 `column-width` 是一种不错的方法，这样可以限制一列中每一行的最小字符数。让我们向内容列中添加一个 9em 的 `column-width`：

```
css/styles.css (excerpt)

#primary article .content,
#tertiary article .content {
  :
  -webkit-column-width: 9em;
  -moz-column-width: 9em;
  column-width: 9em;
}
```

现在，如果在浏览器中调大字号，那么列数会根据保持最小宽度的需要而增加。这保证了可读性，如图 9.6 所示。



图 9.6 以 `em` 为单位声明 `column-width` 可确保每行上具有最少的字符数

9.2.4 columns 简写属性

columns 简写属性是 column-width 和 column-count 属性的复合形式。声明上述两个参数——每列的宽度和列数。

在编写本书时，只有 WebKit 支持这个复合属性，所以需要至少为-moz-实现继续提供独立的属性：

css/styles.css (excerpt)

```
#primary article .content {
  -webkit-columns: 3 9em;
  -moz-column-count: 3;
  -moz-column-width: 9em;
  columns: 3 9em;
}
```

无需为-webkit-和-moz-指定不同的属性，现在可以看到，坚持使用独立的 column-width 和 column-count 属性更简单。当然，决定权在您手上。

9.2.5 列和 height 属性

有了上述声明（没有在元素上指定 height），浏览器将自动平衡列高度，以便每列中的内容在高度上大体相等。

但如果声明了 height 怎么办？当在一个多列块上设置了 height 属性时，在添加新列之前，每一列最多可以增长到该高度。浏览器从第一列开始，创建需要的列数，如果文本极少，则仅创建第一列。最后，如果分配的空间太少，内容将溢出方框外；或者如果设置了 overflow: hidden;，内容将被裁剪。

如果希望在元素上声明 height，但也希望内容分散在多列中，可以使用 column-fill 属性。如果该属性受到支持并设置为 balance，浏览器将平衡列的高度，就像没有声明 height 一样。



边距和空白

即使声明了 height，由于段落上存在边距，列仍然可能没有显示为想要的高度。WebKit 目前会拆分边距和列间的空白，有时在后一列顶部添加额外的空间。Firefox 允许边距超出方框底部，而不是让它们在下一列顶部显示，我们认为这样更有意义。

与 `column-width` 一样,也可能需要以 `em` 而不是像素为单位声明 `height`, 这样,如果用户增加字号,就不太可能裁剪或溢出内容。

9.2.6 其他列功能

除了核心的 `count`、`width` 和 `gap` 属性, CSS3 还为我们提供了其他一些布局多列内容的功能,一些功能还有待支持。

1. `column-rule` 属性

列标尺在本质上是每列之间的边框。`column-rule` 属性指定列标尺的颜色、样式和宽度。该标尺将在列间距之中显示。此属性实际上是 `column-rule-color`、`column-rule-style` 和 `column-rule-width` 属性的简写。

标尺值的语法与 `border` 及相关的 `border-width`、`border-style` 和 `border-color` 属性完全相同。宽度可以为任何长度单位,就像 `border-width` 一样,包含关键字 `medium`、`thick` 和 `thin`。颜色可以是任何支持的颜色值:

```
css/styles.css (excerpt)
-webkit-column-rule: 1px solid #CCCCCC;
-moz-column-rule: 1px solid #CCCCCC;
column-rule: 1px solid #CCCCCC;
```

2. 分列符

有 3 种分列属性可供开发人员用于定义分列符应该出现在何处。`break-before`、`break-after` 和 `break-inside` 属性接受有限数量的关键字作为值,来定义分列符能够和分别应该在元素之前、之后还是内部存在。没有应用到我们定义主要列属性的相同元素,它们应用到嵌入该元素内的其他元素。

可用的值与 CSS 2.1 中的 `page-break-after`、`page-break-before` 和 `page-break-inside` 相同: `auto`、`always`、`avoid`、`left` 和 `right`。CSS3 还为这些属性添加了一些新的可用值: `page`、`column`、`avoid-page` 和 `avoid-column`。`page` 和 `column` 值的作用类似于 `always`,将强制分列。区别在于 `page` 将仅强制分页, `column` 仅应用于列。这在管理换行上提供了更高灵活性。`avoid-page` 和 `avoid-column` 类似,它们的作用类似于 `avoid`。

例如,您可能希望避免在元素中的 `h2` 元素之后立即进行分列。下面给出了实现代码:


```
.columns {
  column-count: 3;
  column-gap: 5px;
}

.columns h2 {
  break-after: avoid;
}
```

目前支持分列的唯一的浏览器引擎是 WebKit。除了需要添加供应商前缀，WebKit 属性还具有与提议的规范中不同的语法（请注意属性名称上添加的单词 column）：

```
-webkit-column-break-after: always;
-webkit-column-break-before: auto;
-webkit-column-break-inside: never;
```

3. 跨越多列

column-span 属性将使元素能够覆盖多列。如果在元素上设置了 column-span: all;，那么在标记中位于该元素之前的所有内容应该位于该元素上方的列中。在标记中该元素之后的列中显示的所有内容应该位于跨区元素下方的列中。

目前，只有 WebKit 支持 column-span（采用 -webkit-column-span 的形式）。因为在不支持时，它会产生差别巨大的外观，所以现在最好避免使用它，除非可以确保所有访问者将使用 WebKit。

例如，对于 The HTML5 Herald 上的第一篇文章，我们可以对 .content div 以外的 article 元素应用列属性，使用 column-span 确保视频覆盖文章的完整宽度。但是，这在支持列但不支持跨区的浏览器（比如 Firefox）中的显示可能很糟糕，所以我们选择保持视频与列内容分开。

9.2.7 其他考虑因素

如果一直练习了我们的所有示例，可能会注意到一些文本块之间存在奇怪的空白，类似于图 9.7。

在非常窄的列中为文本设置了 text-align: justify; 时就会发生此问题，就像我们在 The HTML5 Herald 中所做的一样。这是因为浏览器不知道如何以文字处理程序所做的方式连接单词，所以它们使用空白奇怪地将单词分开，以确保左边和右边保持整齐。

| | |
|--|--|
| Maecenas quis tortor arcu. Vivamus rutrum nunc non neque consectetur quis placerat neque lobortis. Nam vestibulum, arcu sodales feugiat consectetur, nisl orci bibendum elit, eu eiusmod magna sapien ut nibh | sapien ut nibh Nunc eu ullamcorper orci. Quisque eget odio ac lectus vestibulum faucibus eget in metus. In pellentesque faucibus vestibulum. Nulla at vestibulum Nulla at Vivamus rutrum nunc non neque consectetur quis placerat neque lobortis. Nam vestibulum, arcu sodales feugiat consectetur, nisl orci bibendum elit, eu eiusmod magna sapien ut nibh |
| Nunc eu ullamcorper orci. Quisque eget odio ac lectus vestibulum, arcu sodales feugiat consectetur, nisl orci bibendum elit, eu eiusmod magna sapien ut nibh | Nunc eu ullamcorper orci. Quisque eget odio ac lectus vestibulum, arcu sodales feugiat consectetur, nisl orci bibendum elit, eu eiusmod magna sapien ut nibh |
| Vivamus rutrum nunc non neque consectetur quis placerat neque lobortis. Nam vestibulum, arcu sodales feugiat consectetur, nisl orci bibendum elit, eu eiusmod magna sapien ut nibh | Nunc eu ullamcorper orci. Quisque eget odio ac lectus vestibulum, arcu sodales feugiat consectetur, nisl orci bibendum elit, eu eiusmod magna sapien ut nibh |

图 9.7 在列太窄时文本之中可能出现“河流”

对于 The HTML5 Herald, 我们使用了一个名为 Hyphenator¹的 JavaScript 库来连接单词, 保持文本看起来整齐。但是, 这对于您的网站没有必要——我们的列非常窄, 因为我们试图再现旧样式的报纸。很少有真正的网站需要这么窄的整齐的列, 但是如果遇到此问题, 知道存在可用的解决方案也很不错。

9.2.8 渐进增强

尽管列仍然仅有有限的浏览器支持, 但将它们包含在浏览器中也没什么危害, 除非设计人员吹毛求疵。列可以视为一种渐进增强: 使较长的行更容易读取。拥有缺乏列支持的浏览器的人将无法知道它们缺少了什么。例如, 在 Internet Explorer 9 中查看时, The HTML5 Herald 将不包含列, 如图 9.8 所示, 但网站没有被破坏, 只是为了适应浏览器的功能。

VIDEO IS THE FINAL FRONTIER, AND NOW WE HAVE CONQUERED IT!

Aliquam erat volutpat. Mauris vel neque sit amet nunc gravida congue sed sit amet purus. Quisque lacus quam, egestas ac tincidunt a, lacina vel velit. Morbi ac commodo nulla

In condimentum orci id nisl volutpat bibendum. Quisque commodo hendrerit lorem quis egestas. Vivamus rutrum nunc non neque consectetur quis placerat neque lobortis. Nam vestibulum, arcu sodales feugiat consectetur, nisl orci bibendum elit, eu eiusmod magna sapien ut nibh. Aliquam erat volutpat. Mauris vel neque sit amet nunc gravida congue sed sit amet purus

图 9.8 我们的网站没有列——但不影响阅读

¹ <http://code.google.com/p/hyphenator/>

但是，如果列是设计的重要功能并且必须提供给所有访问者，可以使用脚本来提供帮助，比如 Adam Wulf 编写的 jQuery 插件 Columnizer¹。

9.3 媒体查询

现在，我们向 The HTML5 Herald 添加了许多 CSS3 增强。在此过程中，我们通过演示不属于示例网站范围的 CSS3 方面，填补了一些知识空白。所以尽管我们介绍的是列的主题，也可以介绍另一项 CSS3 功能，该功能受到了以各种设备上的受众为目标的设计人员的广泛关注。

在第 1 章中，我们介绍了移动设备的增长率和考虑移动用户的需要的重要性。使用 CSS3 媒体查询，可以实现此目的——创建一个可调整大小来适应不同屏幕分辨率的布局。

媒体查询处于一种称为**自适应 Web 设计**的最新设计趋势的核心。可以依据用户浏览器的功能和尺寸，设计和编码所有页面元素（包括图像和小部件）以无缝且优雅地调整大小和重新对齐。

9.3.1 什么是媒体查询

在 CSS3 之前，开发人员可以使用 media 属性为样式表指定一种媒体类型。所以您可能遇到过类似下面这样的 link 元素：

```
<link rel="stylesheet" href="print.css" media="print">
```

请注意，media 类型指定为 print。除了 print，可接受的值包括 screen、handheld、projection、all 和其他许多很少看到的值（如果有）。media 属性可用于基于查看网站的设备类型，指定加载哪个样式表。这已经成为提供打印样式表的一种非常常见的方法。

根据 W3C 规范，使用 CSS3 的媒体查询，可以“允许为样式表添加更准确的标签，扩展媒体类型的功能。”这是结合使用媒体类型与检查特定媒体功能存在与否的表达式来完成的。一些媒体查询可用针对众多设备更改内容的表示（CSS），而无需更改内容本身（HTML）。

¹ <http://welcome.totheinter.net/columnizer-jquery-plugin/>

9.3.2 语法

我们使用上面的示例，实现一个简单的媒体查询表达式：

```
<link rel="stylesheet" href="style.css" media="screen and (color)">
```

这告诉浏览器，相关的样式表应该用于所有彩色屏幕设备。这很简单，而且应该涵盖了几乎所有受众。可以使用@import 实现相同目的：

```
@import url(color.css) screen and (color);
```

此外，可以使用@media @规则实现媒体查询，本章前面在探讨@font-face 时已介绍过该规则。@media 可能是媒体查询最著名的用法，也是您可能用得最多的方法：

```
@media handheld and (max-width: 380px) {
  /* styles go here */
}
```

在上面的示例中，表达式将应用于最大显示屏宽度为 380 像素的所有手持设备。该代码块中的任何样式将仅应用于与该表达式匹配的设备。

以下是其他一些使用@media 的媒体查询示例，从中可以了解到表达式可以非常灵活和多样。此样式将仅应用于最小设备宽度（或屏幕宽度）为 320 像素且最大设备宽度为 480 像素的设备：

```
@media only screen and (min-device-width: 320px) and
➤(max-device-width: 480px) {
  /* styles go here */
}
```

下面是一个稍微复杂一点的示例：

```
@media only screen and (-webkit-min-device-pixel-ratio: 1.5),
➤only screen and (min-device-pixel-ratio: 1.5) {
  /* styles go here */
}
```

在上面的示例中，我们使用了 only 关键字以及 and 关键字，还有一个逗号（它的行为类似于 or 关键字）。此代码将专门针对 iPhone 4 的较高分辨率显示屏，如果

希望设备显示一组不同的图像，这可能很方便。

9.3.3 媒体查询的灵活性

使用上述语法，媒体查询可用来基于广泛的环境因素更改网站或应用程序的布局。例如，网站使用一种两列布局，可以指定侧栏列位于底部和/或为水平方向，或者可以在较小分辨率的屏幕上完全删除它。在智能电话等小型设备上，可以提供一种完全不同的样式表来删除所有不必要的内容。

此外，可以更改图像和其他通常与用户设备或屏幕分辨率不符的元素的大小。此灵活性可用于为几乎任何类型的设备自定义用户体验，同时保持最重要的信息和网站品牌可供所有用户访问。

9.3.4 浏览器支持

媒体查询得到了不错的浏览器支持：

■ Internet Explorer 9 及更高版本；

■ Firefox 3.5 及更高版本；

■ Safari 3.2 及更高版本；

■ Chrome 8 及更高版本；

■ Opera 10.6 及更高版本；

■ iOS 3.2 及更高版本；

■ Opera Mini 5 及更高版本；

■ Opera Mobile 10 及更高版本；

■ Android 2.1 及更高版本。

唯一存在问题的区域是 Internet Explorer 的早期版本。有两个选项可用于处理此问题：可为这些版本提供没有使用媒体查询的“默认”样式表，提供一种适合大部分屏幕尺寸的布局，或者可以使用基于 JavaScript 的填充内容。这样一种可供使用的解决方案可在 <http://code.google.com/p/css3-mediaqueries-js/> 上找到。

因此，通过利用 CSS3 媒体查询，可以轻松创建一种适用于几乎所有设备和平台的强大方式。

9.3.5 其他阅读材料

在本书中，我们无法描述媒体查询的方方面面。该主题本身就可以写另一本书，而且是一本重要的书。但是如果希望进一步了解媒体查询，一定要查阅以下文章：

- A List Apart 上的 Responsive Web Design¹；
- Smashing Magazine 上的 How to Use CSS3 Media Queries to Create a Mobile Version of Your Site²；
- Cloud Four 博客上的 CSS Media Query for Mobile is Fool's Gold³提供了一种更重要的视角。

9.4 小结

我们现在介绍了 The HTML5 Herald 中应用的所有新 CSS 功能，以及许多没应用到其中的功能。尽管未能涵盖 CSS3 的所有功能，但我们掌握了如今可以使用的多项技术，一些技术在不久的将来应该也很有用。请记住查阅规范（因为这些功能都可能变更）和随时关注浏览器支持的状态。各项功能正在迅速变化，这对于 Web 开发人员而言既大有裨益，又是一项额外的责任。

接下来，我们将介绍一些新的 JavaScript API。虽然我们已经提到，这些内容严格来讲不属于 HTML5 或 CSS3 的范畴，但是人们在谈论这些新技术时常常也会谈到它们。此外，它们非常有趣，所以为什么不了解一下呢？

¹ <http://www.alistapart.com/articles/responsive-web-design/>

² <http://www.smashingmagazine.com/2010/07/19/how-to-use-css3-media-queries-to-create-a-mobile-version-of-your-website/>

³ <http://www.cloudfour.com/css-media-query-for-mobile-is-fools-gold/>

地理定位、离线 Web 应用和 Web 存储

严格地讲，大部分通常被认为是 HTML5 一部分的内容根本就不属于 HTML，而只是一个附加的 API 集，提供各种工具来使我们的网站看起来更好。我们在第 1 章中介绍了 API 的概念，这里将快速复习一下：API 是应用程序编程接口。API 不是一个用户单击按钮即可发生某种操作的可视接口，而是为代码提供一个可供单击的虚拟“按钮”，然后调用一种方法来访问一组功能。在本章中，我们将介绍一些最有用的 API，并简单介绍一下其他 API，使您可以了解更多内容。

有了这些 API，我们可以查找访问者的当前位置，使我们的网站脱机可用，在线更快地执行，并存储有关我们的 Web 应用程序状态的信息，这样当用户返回我们的网站时，他们就可以从离开的位置继续进行操作。



警告

如您所知，P 在 API 中表示编程，因此在接下来的两章中会有一些 JavaScript 代码。如果您刚接触 JavaScript，不用担心！我们会使用带有详细介绍的简单示例尽力帮助您了解如何使用这些新功能。我们假设您已经具备基础知识，但是 JavaScript 是个非常大的主题，要了解更多信息，SitePoint 出版社出版的由 Kevin

Yank 和 Cameron Adams 编写的《Simply JavaScript》是初学者开始的绝佳资源¹。您可能还会发现 Mozilla Developer Network 的《JavaScript Guide》也很有用²。

正如目前在本书中使用的所有 JavaScript 示例一样，为了使这些示例尽可能地简洁且可读，我们使用了 jQuery 库。我们想要演示 API 本身，而不是编写跨浏览器的 JavaScript 代码的复杂性。再次说明，如果您喜欢，可以使用纯 JavaScript 轻松地编写这些代码。

10.1 地理定位

我们介绍的第一个新 API 是地理定位。地理定位允许您的访问者共享其当前位置。

根据访问者访问网站的方式，可以通过下列内容之一确定它们的位置：

- IP 地址；
- 无线网络连接；
- 手机信号接收塔；
- 设备上的 GPS 硬件。

使用上述哪种方法将取决于浏览器和设备的功能。然后，浏览器确定位置并将其传输回地理定位 API。要注意的一点是，正如 W3C 地理定位规范所述：“无法保证 API 会返回设备的实际位置。”³

下列浏览器支持地理定位：

- Safari 5 及更高版本；
- Chrome 5 及更高版本；
- Firefox 3.5 及更高版本；
- Internet Explorer 9 及更高版本；
- Opera 10.6 及更高版本；

¹ 墨尔本：SitePoint，2007

² <https://developer.mozilla.org/en/JavaScript/Guide>

³ <http://dev.w3.org/geo/api/spec-source.html#introduction>

- iOS (Mobile Safari) 3.2 及更高版本;
- Android 2.1 及更高版本。

10.1.1 隐私问题

并不是每个人都想与您共享他的位置，因为这涉及隐私问题。因此，您的访问者可以选择性地共享他们的位置。除非用户同意，否则不会将任何信息传输到您的网站或 Web 应用程序。

可通过浏览器顶部的提示做出决定。图 10.1 显示了 Chrome 中的提示。

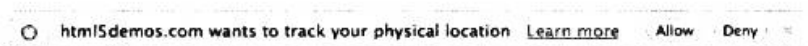


图 10.1 地理定位用户提示



阻止 Chrome 中的地理定位提示

注意，如果您正在从本地查看页面，而不是从内部服务器查看页面，那么 Chrome 可能会阻止您的网站显示此提示。出现这种情况时，您将在地址栏中看到一个图标告诉您阻止了此提示。

目前，还没有方法可以解决这种问题，但是您可以在其他浏览器中测试您的功能，或者将代码部署到测试服务器（可以是您机器上的本地服务器、虚拟机或者是实际 Internet 服务器）上。

10.1.2 地理定位方法

使用地理定位，您可以确定用户的当前位置，还可以向您通知他们的位置变化，例如，提供实时驾驶路线的 Web 应用程序。

可以通过地理定位 API 中目前可用的三种方法来控制这些不同的任务。

- `getCurrentPosition`;
- `watchPosition`;
- `clearPosition`。

我们将重点介绍第一种方法 `getCurrentPosition`。

10.1.3 使用 Modernizr 检查支持

在我们试图使用地理定位之前，应确保访问者的浏览器支持它。我们可以使用 Modernizr 进行这一工作。

首先创建一个名为 `determineLocation` 的函数。我们已经将该函数放入了 JavaScript 文件 `geolocation.js` 中，并将该文件包含在我们的页面中。

在该函数内，我们首先使用 Modernizr 来检查是否支持地理定位：

```
geolocation.js (excerpt)
```

```
function determineLocation() { ❶
    if (Modernizr.geolocation) { ❷
        navigator.geolocation.getCurrentPosition(displayOnMap);
    }
    else {
        // geolocation is not supported in this browser
    }
}
```

我们来逐行检查一下这些代码：

- ❶ 我们声明了一个名为 `determineLocation` 的函数，以包含我们的位置检查代码。
- ❷ 我们检查了 Modernizr 对象的 `geolocation` 属性，以查看在当前浏览器中是否支持地理定位。有关 Modernizr 对象的工作方式的更多信息，请参见附录 A。如果支持地理定位，就继续检查第三行，位于 `if` 语句中。如果不支持地理定位，我们可以检查 `else` 语句中的代码。

我们假设支持地理定位。

10.1.4 获取当前位置

`getCurrentPosition` 方法可以具有一个、两个或三个参数。下面是该方法的定义概述（来自 W3C 的地理定位 API 规范）：¹

```
void getCurrentPosition(successCallback, errorCallback, options);
```

¹ <http://dev.w3.org/geo/api/spec-source.html>

在这 3 个参数中，只有第一个参数 `successCallback` 是必需的。`successCallback` 是在位置确定时您想要调用的函数的名称。

在我们的示例中，成功找到位置后，新的 `Position` 对象将调用 `displayOnMap` 函数。此 `Position` 对象将包含设备的当前位置。



回调

回调是将一个函数作为参数传递给另一个函数。父函数完成后将执行回调。在 `getCurrentPosition` 这种情况下，`successCallback` 仅会在 `getCurrentPosition` 完成并确定了位置时运行一次。

10.1.5 地理定位的 `Position` 对象

下面将详细介绍 `Position` 对象。正如在地理定位 API 中定义的那样，`Position` 对象有两个属性：一个包含位置的坐标 (`coords`)，另一个包含位置确定时的时间戳 (`timestamp`)：

```
interface Position {
    readonly attribute Coordinates coords;
    readonly attribute DOMTimeStamp timestamp;
};
```



接口

与上文一样，HTML5、CSS3 和相关规范包含很多“接口”。起初看起来它们很吓人，但是不要担心。它们只是进入某种属性、方法或对象的内容的概述。大多数时间，含义都很明确。如果含义不明确，那么它们总是会附带一些属性的文字描述。

但是经度和纬度存储在哪里呢？它们存储在 `Coordinates` 对象中。`Coordinates` 对象也是在 W3C 地理定位规范中定义的。下面是它的属性：

```
interface Coordinates {
    readonly attribute double latitude;
    readonly attribute double longitude;
    readonly attribute double? altitude;
    readonly attribute double accuracy;
```

```

    readonly attribute double? altitudeAccuracy;
    readonly attribute double? heading;
    readonly attribute double? speed;
};

```

其中一些属性中 `double` 的后面有一个问号，该问号表示不保证会有该属性。如果浏览器无法获得这些属性，那么它们的值将是 `null`。例如，极少数的计算机或智能手机具有高度计，因此，大多数时间，您不会从地理定位调用得到 `altitude` 值。一定会有的三个属性是 `latitude`、`longitude` 和 `accuracy`。

`latitude` 和 `longitude` 的含义是不言而喻的，为您提供您想要的：用户的经度和纬度。`accuracy` 属性以米为单位，告诉您经度和纬度信息的精确度。

`altitude` 属性是以米为单位的海拔，而 `altitudeAccuracy` 属性是海拔的精确度，也以米为单位。

我们跨多个位置跟踪用户时，`heading` 和 `speed` 属性是唯一相关的。如果我们正在提供实时骑行或驾驶路线，那么这些属性将非常重要。`heading`（如果存在）将告诉我们用户移动的方向与正北之前的角度。`speed`（如果存在）告诉我们用户移动的速度，单位为米每秒。

10.1.6 获取经度和纬度

我们的 `successCallback` 被设置为函数 `displayOnMap`。下面是此函数的外观：

geolocation.js (excerpt)

```

function displayOnMap(position) {
    var latitude = position.coords.latitude;
    var longitude = position.coords.longitude;
    // Let's use Google Maps to display the location
}

```

函数的第一行从 `Position` 对象获取 `Coordinates` 对象，前者由 API 传递给我们的调用。在 `Coordinates` 对象内是 `latitude` 属性，该属性存储在名为 `latitude` 的变量中。我们对 `longitude` 执行同样的操作，将其存储在名为 `longitude` 的变量中。

为了在地图上显示用户的位置，我们将利用 Google Maps JavaScript API。但在

我们使用此 API 之前,我们需要在 HTML 页面中为其添加一个引用。无需下载 Google Maps JavaScript 库并将其存储在我们的服务器上,我们可以指向 Google 公开可用的 API 版本:

```

                                geolocation.js (excerpt)
:
<!-- google maps API -->
<script type="text/javascript" src="http://maps.google.com/maps/
api/js?sensor=true">
</script>
</body>
</html>

```

Google Maps 有一个 `sensor` 参数,表示此应用程序是否使用传感器(GPS 设备)来确定用户的位置。您可以在上面的示例中看到 `sensor=true`。您必须将此值明确设置为 `true` 或 `false`。因为 W3C 地理定位 API 无法得知获得的信息是否来自传感器,对于大多数 Web 应用程序来说,设置为 `false` 都是安全的,除非它们是专为具有 GPS 功能的设备创建的,比如 iPhone。

10.1.7 加载地图

既然我们已经包含了 Google Maps JavaScript,那么首先我们需要为页面添加一个元素以包含地图,然后为用户提供一种方式来通过单击按钮调用我们的 `determine Location` 方法。

要完成第一步,我们应该在 The HTML5 Herald 侧栏的第三个广告框下创建第四个框。我们将它封装在 `article` 元素中,正如我们对所有广告进行的操作一样。在该元素中,我们将创建一个名为 `mapDiv` 的 `div`,将其作为地图的占位符。还要添加一个标题来告诉用户我们试图查找的内容:

```

                                index.html (excerpt)
<article id="ad4">
  <div id="mapDiv">
    <h1>Where in the world are you?</h1>
    <form id="geoForm">
      <input type="button" id="geobutton" value="Tell us!">
    </form>
  </div>
</article>

```

我们还为此新 HTML 添加了一些样式：

```
css/styles.css (excerpt)

#ad4 h1 {
    font-size: 30px;
    font-family: AcknowledgementMedium;
    text-align: center;
}

#ad4 {
    height: 140px;
}

#mapDiv {
    height: 140px;
    width: 236px;
}
```

图 10.2 显示了新侧栏框的外观。



图 10.2 支持用户告诉我们其位置的新小部件

第二步是在我们单击按钮时调用 `determineLocation`。使用 jQuery，可以轻松地将我们的函数附加到按钮的单击事件中：

```
js/geolocation.js (excerpt)

$('document').ready(function(){
    $('#geobutton').click(determineLocation);
});
```



文档就绪

在上述代码段中,第二行完成了主要工作。`$('#document').ready(function(){ ... });`片段只是告诉 jQuery 在页面完全载入之前不要运行我们的代码。这是必要的,因为我们的代码可能会在 `#geobutton` 元素存在之前寻找它,从而产生错误。

这在 JavaScript 和 jQuery 中是非常普遍的模式。如果您刚开始前端编程,请相信我们,您会学到很多内容。

有了此代码,无论何时只要单击该按钮就会调用 `determineLocation`。

现在,让我们返回 `displayOnMap` 函数,并了解实际显示地图的具体细节。首先,我们将创建一个 `myOptions` 变量来存储将传递到 Google Map 的一些选项:

js/geolocation.js (excerpt)

```
function displayOnMap(position) {
    var latitude = position.coords.latitude;
    var longitude = position.coords.longitude;

    // Let's use Google Maps to display the location
    var myOptions = {
        zoom: 14,
        mapTypeId: google.maps.MapTypeId.ROADMAP
    };
}
```

我们将设置的第一个选项是缩放级别。对于完整地图,使用缩放级别 0。缩放级别越高,您距离位置就越近,画面(或窗口)就越小。我们使用缩放级别 14 来放大查看街道。

我们将设置的第二个选项是显示的地图类型。我们可以从下列选项进行选择:

- `Google.maps.MapTypeId.ROADMAP;`
- `Google.maps.MapTypeId.SATELLITE;`
- `Google.maps.MapTypeId.HYBRID;`
- `Google.maps.MapTypeId.TERRAIN.`

如果您之前使用过 Google Maps 网站，您将熟悉这些地图类型。ROADMAP 是默认设置，而 SATELLITE 向您显示逼真的图片。HYBRID 是 ROADMAP 和 SATELLITE 的组合，而 TERRAIN 将显示高地和水等元素。我们将使用默认设置 ROADMAP。



Google Maps 中的选项

要了解有关 Google Maps 选项的更多信息，请参见 Google Maps 教程的 Map Options 部分¹。

现在已经设置了我们的选项，是时候创建地图了！我们通过创建带有 `new google.maps.Map()` 的新 Google Maps 对象来实现这一目的。

我们传递的第一个参数是 DOM 方法 `getElementById` 的结果，我们用它来获取放置在 `index.html` 页面中的占位符 `div`。将此方法的结果传递到新 Google Maps，这意味着创建的地图将放置在该元素中。

我们传递的第二个参数是刚刚设置的选项集合。我们将生成的 Google Maps 对象存储在名为 `map` 的变量中：

js/geolocation.js (excerpt)

```
function displayOnMap(position) {
    var latitude = position.coords.latitude;
    var longitude = position.coords.longitude;

    // Let's use Google Maps to display the location
    var myOptions = {
        zoom: 16,
        mapTypeId: google.maps.MapTypeId.ROADMAP
    };

    var map = new google.maps.Map(document.getElementById("mapDiv"),
    ↪myOptions);
```

现在我们已经有了地图，让我们添加一个标记来表示用户的位置。标记是我们在标示位置的 Google Maps 上看到的小红色水滴。

为了创建一个新的 Google Maps 标记对象，我们需要将它传递给另一个对象类型：`google.maps.LatLng` 对象，它只是一个表示经度和纬度的容器。通过调用

¹ <http://code.google.com/apis/maps/documentation/javascript/tutorial.html#MapOptions>

`google.maps.LatLng` 并将它作为参数传递给 `latitude` 和 `longitude` 变量，第一个新行创建了新的 Google Maps 标记对象。

现在已经有了 `google.maps.LatLng` 对象，我们可以创建一个标记。我们调用 `new google.maps.Marker`，然后在两个大括号（{}）之间将 `position` 设置为 `LatLng` 对象，将 `map` 设置为 `map` 对象，将 `title` 设置为 "HelloWorld!"。标题是我们将鼠标悬停在标记上时显示的内容。

js/geolocation.js (excerpt)

```
function displayOnMap(position) {
    var latitude = position.coords.latitude;
    var longitude = position.coords.longitude;

    // Let's use Google Maps to display the location
    var myOptions = {
        zoom: 16,
        mapTypeId: google.maps.MapTypeId.ROADMAP
    };

    var map = new google.maps.Map(document.getElementById("mapDiv"),
    ↪myOptions);

    var initialLocation = new google.maps.LatLng(latitude, longitude);

    var marker = new google.maps.Marker({
        position: initialLocation,
        map: map,
        title: "Hello World!"
    });
}
```

最后一步是在初始点将地图居中，我们通过调用具有 `LatLng` 对象的 `map.setCenter` 来实现这一操作：

```
map.setCenter(initialLocation);
```

您可以在在线文档中找到大量关于 Google Maps JavaScript API 第 3 版的文档¹。

¹ <http://code.google.com/apis/maps/documentation/javascript/>

10.1.8 关于旧式移动设备的结束语

虽然当前的移动设备浏览器能很好地支持 W3C 地理定位 API,但是您可能需要在旧式移动设备中支持它,并支持所有可用的地理定位 API。如果是这样,您应该查看开源库 `geo-location-javascript`¹。

10.2 离线 Web 应用

现在,越来越多在旅途中的人会访问我们的网站。其中有很多人一直都在使用移动设备,假设我们的访问者总是具有实时的网络连接是不明智的,如果访问者在脱机时也能浏览我们的网站或者是使用我们的 Web 应用程序,那该有多好!幸运的是,使用离线 Web 应用确实能够实现此目的。

HTML5 的离线 Web 应用允许我们在脱机时与网站进行互动。起初这听起来似乎自相矛盾:根据定义,Web 应用程序只存在于网络上。但是,越来越多的基于网络的应用程序受益于离线使用。您可能使用一个基于网络的电子邮件客户端,比如 Gmail;如果您在上班途中的地铁上,能够在应用程序中撰写草稿,那岂不是非常有用?关于网上待办事项,联系管理员,或是 office 应用程序?所有这些应用程序的示例,都受益于连接网络,但是在隧道中网络连接信号被切断的情况下,我们仍想继续使用。

下列浏览器支持离线 Web 应用规范:

- Safari 4 及更高版本;
- Chrome 5 及更高版本;
- Firefox 3.5 及更高版本;
- Opera 10.6 及更高版本;
- iOS (Mobile Safari) 2.1 及更高版本;
- Android 2.0 及更高版本。

¹ <http://code.google.com/p/geo-location-javascript/>

10.2.1 工作原理：HTML5 应用程序缓存

离线 Web 应用通过利用应用程序缓存运行。应用程序缓存可以存储整个离线网站：所有的 JavaScript、HTML 和 CSS，以及所有的图像和资源。

这听起来很不错，但是您可能想知道，如果发生变化，那会出现什么情况呢？这就是应用程序缓存的美妙之处：每次用户在线访问页面时，您的应用程序都会自动更新。在您的文件中，即使是一个字节的数据变化，应用程序缓存都将重新加载该文件。



应用程序缓存与浏览器缓存

浏览器保留了自己的缓存以加快网站加载的速度；但是，这些缓存仅用于避免重新加载指定文件——在网络连接的情况下。浏览器会缓存了一个页面的所有文件。如果试图离线单击一个链接，您会收到一条错误消息。

使用离线 Web 应用，我们能够告诉浏览器应从网络上缓存或获取哪些文件，并且在缓存失败的时候，应回退到哪里。它使我们能够更好地控制如何缓存我们的网站。

10.2.2 设置站点离线工作

设置离线 Web 应用程序有以下三个步骤。

- (1) 创建一个 `cache.manifest` 文件。
- (2) 确保清单文件具有正确的内容类型。
- (3) 所有 HTML 文件都指向 **cache manifest**。

The HTML5 Herald 根本不是一个真正的应用程序，所以它不是那种您想要提供离线功能的网站。但是，它非常简单，而且没有真正的缺点，所以我们将介绍使其离线可用的步骤，看一看它的工作原理。

1. `cache.manifest` 文件

虽然名称奇特，但是 `cache.manifest` 文件只是一个遵循一定格式的文本文件。

以下是一个简单的 `cache.manifest` 文件示例：

```
CACHE MANIFEST
CACHE:
index.html
photo.jpg
main.js

NETWORK:
*
```

`cache.manifest` 文件的第一行代码必须显示为 `CACHE MANIFEST`。在这行后面，我们输入 `CACHE:`，然后列出我们将存储在访问者硬盘的所有文件。此 `CACHE:` 部分也被称为显式部分（我们明确告诉浏览器缓存这些文件）。

使用 `cache.manifest` 文件第一次访问页面时，访问者的浏览器将会在本地图复制一份在本部分定义的所有文件。在以后的访问中，浏览器将加载文件的本地副本。

列出了所有想要离线储存的文件后，我们可以指定**在线白名单**。这里，我们定义了所有不离线储存的文件，因为通常它们的内容需要互联网访问才具有实际意义。例如，您可能有一个 PHP 脚本 `lastTenTweets.php`，此文件捕捉 Twitter 最后 10 分钟的更新，并将它们显示在 HTML 页面上。脚本将仅能够获取您在线的最后 10 分钟 Twitter 的信息，所以存储离线页面没有任何意义。

此部分的第一行是 `NETWORK`。当用户在线时，在 `NETWORK` 部分中指定的任何文件总是会重新加载，并在离线时不可用。

下面是**在线白名单**的一个示例：

```
NETWORK
lastTenTweets.php
```

在显式部分中，我们必须煞费苦心地列出所有想要离线存储的文件。与显式部分不同，在**在线白名单**部分中，我们可以使用快捷方式：通配符`*`。这个星号告诉浏览器，应从服务器中获取没有在显式部分（并且因此也不存储在应用程序缓存中）中提到的任何文件或 URL。

下面是一个使用了通配符的**在线白名单**的示例：

```
NETWORK
*
```



所有的 URL 都需说明

网站中的每一个 URL 都必须在 `cache.manifest` 文件中进行说明, 即使只是简单链接的 URL。如果在 `cache.manifest` 文件中没有说明, 即使您在线, 资源或 URL 也将加载失败。为了避免这个问题, 您应在 `NETWORK` 部分中使用 `*`。

您也可以另起新的一行, 以 `#` 开头, 在 `cache.manifest` 文件中添加注释。所有在 `#` 之后的内容都将被忽略。请小心, 不要将注释作为 `cache.manifest` 文件中的第一行, 如前所述, 第一行必须是 `CACHE MANIFEST`。但是, 您可以在其他任何行中添加注释。

在注释中注明 `cache.manifest` 文件的版本号是一个非常好的做法 (我们将在稍后解释其原因):

```
CACHE MANIFEST
# version 0.1
CACHE:
index.html
photo.jpg
main.js

NETWORK:
*
```

2. 在服务器上设置内容类型

为了使您的站点能够离线工作, 我们需要做的下一步是确保您的服务器正确配置以为 `cache.manifest` 文件提供服务。这通过设置由服务器提供的内容类型和 `cache.manifest` 文件来完成。我们在第 5 章的“`MIME 类型`”一节介绍了内容类型, 如果您需要复习一下, 可以跳回到该章节。

假设您使用的是 Apache Web 服务器, 在 `.htaccess` 文件中添加以下代码:

```
AddType text/cache-manifest .manifest
```

3. 将 HTML 指向清单文件

我们所做的最后一步是将 HTML 页面指向清单文件。通过设置每一个页面中的 `html` 元素的 `manifest` 属性来完成这一步:


```
<!doctype html>
<html manifest="/cache.manifest">
```

完成这一步后，就完成了所有步骤！我们的网页现在可以离线工作了。更好的是，由于所浏览的网页的任何内容都没有更改且存储在本地，因此现在网页的加载速度会更快——即使我们的访问者处于在线状态。



为每一页执行相同的操作

网站的每一个 HTML 页面都必须设置 html 元素的 manifest 属性。一定要这样做，否则您的应用程序可能无法存储在应用程序缓存中！在您的整个应用程序中，只能有一个 cache.manifest 文件，Web 应用程序的每一个 HTML 页面都需要 `<html manifest="/cache.manifest">`。

10.2.3 获取离线存储站点的权限

与使用地理定位一样，在网站使用 cache.manifest 文件时，浏览器会提供一个权限提示。与地理定位不同的是，并不是所有浏览器都要求做到这一点。当出现权限提示时，提示将要求用户确认是否将网站设置为离线可用。图 10.3 显示了 Firefox 中的权限提示。



图 10.3 提示允许将离线 Web 应用存储在应用程序缓存中

10.2.4 离线测试

完成了设置离线网站的所有步骤后，我们便可以离线测试网页。Firefox 和 Opera 提供了菜单选项，支持您离线工作，所以无需切断网络连接。如果使用的是 Firefox，那么菜单选择 File>Work Offline，如图 10.4 所示。

虽然从浏览器菜单中选择离线很方便，但是最为理想的是：在测试离线 Web 应用时，完全关闭网络连接。

测试应用程序缓存是否正在存储您的站点

进入离线状态是抽查应用程序缓存是否工作的一种很好的方式，但是，为了更深入的调试，我们需要更好的检测工具。幸运的是，Chrome 的网页审查工具有一些

很好的功能可用于检查应用程序缓存。



图 10.4 使用 Firefox 的 Work Offline 模式测试离线 Web 应用

要检查 `cache.manifest` 文件是否具有正确的内容类型，以 Chrome 浏览器为例，可遵循以下步骤（<http://html5laboratory.com/s/offline-application-cache.html> 有一个您可使用的示例）：

（1）在 Chrome 浏览器中，导航到主页的 URL。

（2）打开 Web Inspector（单击扳手图标，然后选择 `Tools>Developer Tools`）。

（3）打开 Console 选项卡，并查看与 `cache.manifest` 文件相关的任何错误，如果一切运行良好，您应该看到以“Document loaded from Application Cache with manifest”开头并以指向 `cache.manifest` 文件的路径结尾的一行内容。若有任何错误，则都会在 Console 中显示，所以要注意在这里显示的错误和警告。

（4）打开 Resources 选项卡。

（5）展开 Application Cache 部分，会列出您的域（在本例中为 `www.html5laboratory.com`）。

（6）单击域。在右侧列出的应是储存在 Chrome 应用程序缓存中的所有资源，如图 10.5 所示。

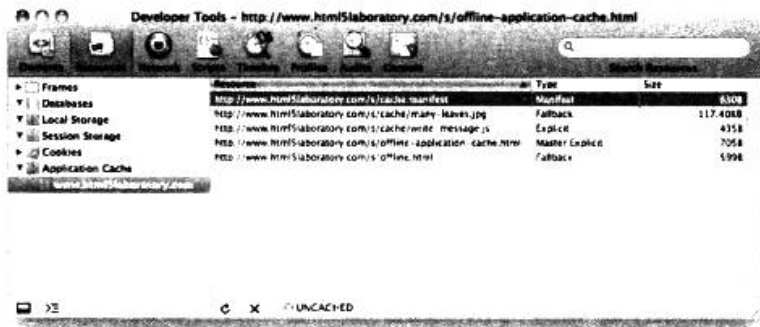


图 10.5 查看储存在 Chrome 应用程序缓存的内容

10.2.5 使 The HTML5 Herald 离线可用

现在，我们了解了网站离线可用的要素，下面让我们以 The HTML5 Herald 为例，练习一下所学习的内容。第一步是创建 `cache.manifest` 文件。您可以使用 Mac 中类似 TextEdit 的程序或者 Windows 中的记事本创建它，但是必须确保文件格式是纯文本。如果您使用 Windows，那么您很幸运！只要您使用记事本创建，它便已经是纯文本了。使用 Mac 的 TextEdit 将文件转换为纯文本，选择 `Format > Make Plain Text`。首先，在文本的顶部包含 `CACHE MANIFEST` 行。

接下来，我们需要在显式部分中添加想要离线使用的所有资源，它以单词 `CACHE:` 开始，我们必须在这部分列出所有文件。由于我们网站上没有任何内容需要网络访问（有一个内容，但是我们稍后再介绍它），因此我们将仅为 `NETWORK` 部分添加一个星号来捕捉在显式部分中有可能漏掉的任何文件。

下面是我们的 `cache.manifest` 文件的摘录：

cache.manifest (excerpt)

```
CACHE MANIFEST
#v1
index.html
register.html

js/hyphenator.js
js/modernizr-1.7.min.js
css/screen.css
css/styles.css
images/bg-bike.png
images/bg-form.png
:
fonts/League_Gothic-webfont.eot
fonts/League_Gothic-webfont.svg
:

NETWORK:
*
```

在文件中添加了所有资源后，将它保存为 `cache.manifest`。一定要将扩展名设置为 `.manifest` 而不是 `.txt` 或其他类型的扩展名。

然后，请设置服务器，以提供具有正确内容类型的清单文件。

最后一步是为两个 HTML 页面中的 `html` 元素添加 `manifest` 属性。

我们为 `index.html` 和 `register.html` 都添加了 `manifest` 属性，代码如下：

```
<!doctype html>  
<html lang="en" manifest="cache.manifest">
```

好，我们设置完成了！现在，无论我们是否有网络连接，都可以在闲暇时浏览 The HTML5 Herald 了。

10.2.6 离线 Web 应用存储的限制

离线 Web 应用规范没有为应用程序缓存定义一个具体的存储限制，它规定浏览器应创建并执行存储限制。作为一般原则，假设您的工作空间不超过 5MB，这是一个好主意。

我们指定的离线存储的几个文件是视频文件。根据视频文件的大小，由于他们可能超过了浏览器的存储限制，因此提供离线使用可能没有任何意义。

在这种情况下，我们能做些什么呢？我们可以将大视频文件放到 `NETWORK` 部分中，然后，在浏览器试图离线播放视频时，用户将仅简单地看到一个令人不愉快的错误。

一种更好的替代方案是使用 `cache.manifest` 的可选部分：后备部分。

10.2.7 后备部分

这部分允许我们定义在资源加载失败时用户所看到的信息。在 The HTML5 Herald 示例中，并不是存储视频文件并将它放到显式部分中，而是利用后备部分更有意义。

后备部分中的每一行都需要两个条目。第一个是想要提供后备内容的文件。您可以指定一个具体文件，或使用部分路径名，比如 `media/`，它将指向位于 `media` 文件夹中的任何文件。第二个条目是指定的文件加载失败时您想要显示的信息。

如果无法加载文件，我们可以加载一个影片的第一帧的静止图像。我们使用部分路径名 `media/` 来同时为两个视频文件定义后备。

cache.manifest (excerpt)

```
FALLBACK:
media/ images/ford-plane-still.png
```

当然，这有一点多余，正如您在第 5 章所了解的，HTML5 的 video 元素已经包含了后备图片，将在视频加载失败时显示。

所以，为了使用这个概念进行更多的练习，我们添加另一个后备。在任何页面都不能加载的情况下，定义一个后备文件告诉您站点已离线，这是一个非常好的做法。我们可以创建一个简单的 **offline.html** 文件：

offline.html

```
<!doctype html>
<html lang="en" manifest="/cache.manifest">
  <head>
    <meta charset="utf-8">
    <title>We are offline!</title>
    <link rel="stylesheet" href="css/styles.css?v=1.0"/>
  </head>
  <body>
    <header>
      <h1>Sorry, we are now offline!</h1>
    </header>
  </body>
</html>
```

现在，在缓存清单的后备部分，我们可以指定/，它将匹配站点的任何页面。如果页面加载失败或不在应用程序缓存中，我们将跳回 **offline.html** 页面：

cache.manifest (excerpt)

```
FALLBACK:
media/ images/video-fallback.jpg
/ /offline.html
```



Safari 离线应用程序缓存加载媒体文件失败

目前，在 Safari 5 中有一个 bug，像 **.mp3** 和 **.mp4** 这样的媒体文件不能从离线应用程序缓存加载。

10.2.8 刷新缓存

当使用缓存清单时，在显式部分中指定的文件直到接到进一步通知时才被存入缓存。在开发时，这可能会引起一些问题：您可能更改了文件，但不能看到页面所反映出的变化，这可能让您很苦恼。

更重要的是，将文件上传到在线网站上，您可能希望通过一种方式告诉浏览器更新它们的应用程序缓存。可以通过修改 **cache.manifest** 文件来做到这一点。当浏览器加载已经含有 **cache.manifest** 文件的站点时，它将会检查是否更改了清单文件。如果没有，它将认为现有的应用程序缓存就是需要运行的应用程序，所以也就不再下载任何东西。如果 **cache.manifest** 文件发生改变，浏览器将通过重新下载所有的指定文件，重新构建应用程序缓存。

这就是为什么要在 **cache.manifest** 的注释中指定版本号的原因。这样，即使文件清单完全一样，我们仍然可以向浏览器表明应更新应用程序缓存；我们只需递增版本号就可以了。

存储缓存

这可能听起来有些荒唐，但是浏览器可能会缓存您的 **cache.manifest** 文件。您可能会问，为什么？这是由于 HTTP 处理缓存的方式。

为了加速网页的整体性能，根据通过 HTTP 规范制定的原则¹，由浏览器完成缓存。那么您需要了解这些原则的哪些方面呢？浏览器接收到某些 HTTP 头，其中包括 Expire 头。这些 Expire 头将告诉浏览器缓存中的文件的过期时间，以及需要从服务器更新的时间。

如果您的服务器为清单文件提供了缓存的指令（这通常是静态文件的默认状态），那么浏览器将非常高兴地使用文件的缓存版本，而无需从服务器中获取更新的版本。由于它认为清单没有被更改，因此将不会重新下载任何应用程序文件！

如果您发现不能够使浏览器刷新其应用程序缓存，就尝试清除常规的浏览器缓存。您也可以更改服务器设置，以发送明确的、不缓存 **cache.manifest** 文件的指令。

如果您站点的 Web 服务器运行的是 Apache，可以通过将下列内容添加到 **.htaccess** 文件来告诉 Apache 不要缓存 **cache.manifest** 文件：

¹ <http://www.w3.org/Protocols/rfc2616/rfc2616-sec13.html>

.htaccess (excerpt)

```
<Files cache.manifest>
  ExpiresActive On
  ExpiresDefault "access"
</Files>
```

`<Files cache.manifest>`告诉 Apache 仅应用遵循 **cache.manifest** 文件的原则。`ExpiresActive On` 和 `ExpiresDefault "access"` 的结合使用，总是强制 Web 服务器使缓存中的 **cache.manifest** 到期。这样做的结果是：浏览器将不缓存 **cache.manifest** 文件。

10.2.9 我们在线吗

有时，您需要知道用户是离线浏览网页还是在线浏览网页。例如，在 Web 邮件应用程序中，在线存储草稿会将它发送到服务器并保存到数据库中；但是如果是离线，您可能想要将信息存在本地，然后等到用户在线时，再发送到您的服务器。

离线 Web 应用 API 提供了一些简单的方法和事件对此进行管理。对于 The HTML5 Herald，您可能已经注意到了，在离线时，页面运行得非常好：您可以从主页导航到注册表，播放视频，随意浏览，而没有任何问题。但是，当您试图使用在本章前面创建的地理定位小部件时，事情就不那么妙了。这是有道理的：没有互联网连接，我们的网页就没有办法找到您的位置（除非您的设备具有 GPS 功能），更不用说使用 Google Maps 来检索地图了。

让我们看一下如何解决这个问题。我们将向用户提供一条信息，表明离线时不提供此功能。这确实非常简单：支持离线 Web 应用的浏览器允许您访问 `navigator.online` 属性，如果是在线，它的值为 `true`，否则为 `false`。下面演示了在 `determineLocation` 方法中如何使用它：

js/geolocation.js (excerpt)

```
function determineLocation(){
  if (navigator.onLine) {
    // find location and call displayOnMap
  } else {
    alert("You must be online to use this feature.");
  }
}
```

尝试一下，使用 Firefox 或 Opera，首先导航到页面并单击按钮加载地图。一旦您满意它的运行，选择离线运行，重新加载页面，并再次单击按钮。此时您将接收到一条帮助消息，告诉您需要在线访问地图。

其他一些可能对您有用的功能包括在浏览器上线或下线时触发的事件。这些事件在 window 元素上触发，简称为 window.online 和 window.offline。例如，这些事件可以使您无论是在上线时与服务器同步信息，还是离线时在本地存储数据，都允许脚本对这些状态的改变进行响应。

还有其他一些事件和方法可用于处理应用程序缓存，但是我们在此讨论的这些都是最重要的。它们足以使大部分网站和应用程序能够离线运行。

10.2.10 其他阅读材料

如果您想了解关于离线 Web 应用的更多内容，有以下一些资源可供参考：

- 📖 The WHATWG Offline Web Applications 规范¹；
- 📖 HTML5 Laboratory 的 Using the cache manifest to work offline²；
- 📖 Opera 的 Offline Application Developer's Guide³；
- 📖 Peter Lubbers 的 Slide Share presentation on Offline Web Applications⁴；
- 📖 Mark Pilgrim 的离线 Web 应用练习⁵；
- 📖 Safari 的 Offline Applications Programming Guide⁶。

10.3 Web 存储

Web 存储 API 为用户如何在计算机或设备上本地存储简单的数据定义了标准。在 Web 存储标准出现之前，网络开发人员经常将用户信息存储在 cookie 中，或使用插件。有了 Web Storage，现在对于如何存储高达 5MB 的由网站或 Web 应用程序生成的简单数据有了一个标准的定义。更好的是，Web 存储已经在 Internet Explorer 8

¹ <http://www.whatwg.org/specs/web-apps/current-work/multipage/offline.html#offline>

² <http://www.html5laboratory.com/working-offline.php>

³ <http://dev.opera.com/articles/view/offline-applications-html5-appcache/>

⁴ <http://www.slideshare.net/robinzimmermann/html5-offline-web-applications-silicon-valley-usergroup>

⁵ <http://diveintohtml5.org/offline.html>

⁶ <http://developer.apple.com/library/safari/#documentation/iPhone/Conceptual/SafariJSDatabaseGuide/OfflineApplicationCache/OfflineApplicationCache.html>

中运行了！

由于在您离线工作时，需要在某些地方存储用户数据，并且 Web 存储提供了此项服务，因此，Web 存储是对离线 Web 应用一个极好的补充。

下列浏览器支持 Web 存储：

- Safari 4 及更高版本；
- Chrome 5 及更高版本；
- Firefox 3.6 及更高版本；
- Internet Explorer 8 及更高版本；
- Opera 10.5 及更高版本；
- iOS (Mobile Safari) 3.2 及更高版本；
- Android 2.1 及更高版本。

10.3.1 两种存储

有两种类型的 HTML5 Web 存储：会话存储和本地存储。

1. 会话存储

会话存储可以让我们跟踪特定窗口或选项卡的数据。它支持我们隔离每个窗口的信息。即使用户在两个窗口访问同一个站点，每一个窗口都将有自己独立的会话存储对象，因此会有独立的不同的数据。

会话存储并不是持久的——它仅是在一个特定网站的用户会话所持续的时间（换句话说，就是打开浏览器窗口或选项卡并浏览站点的时间）。

2. 本地存储

与会话存储不一样，本地存储可以使我们通过浏览器在用户的计算机上存储永久的数据。将来当用户再次访问站点时，可以检索所有存储在本地的数据。

试想一下在线购物：对于用户来说，在同一个站点打开多个窗口或选项卡是很平常的事情。例如，您正打算购买一双鞋，并且您想对比两个品牌的价格和用户评论。您可能为每一个品牌都打开一个独立的窗口，但是不管您搜索的是什么品牌或样式的鞋，鞋的号码总是一样的。那么在每一个新窗口重复同样的搜索部分是非常麻烦的。

在这种情况下，本地存储可提供帮助。在每次打开新窗口浏览时，不需要用户再次指定鞋的号码，我们可以将此信息存储在本地存储中。这样，当用户打开新窗口浏览其他品牌和样式时，所搜索出的鞋都是用户所需的同一尺码的鞋。此外，由于我们在用户的计算机上存储了此信息，因此以后当他们再次访问该站点时，将仍能够访问这些信息。



Web Storage 是特定于浏览器的

有一点非常重要，请记住，使用 Web 存储进行工作时，如果用户使用 Safari 访问您的网站，那么所有数据都存储在 Safari 的 Web 存储库中。如果用户使用 Chrome 再次访问该站点，将不能够使用通过 Safari 存储的数据。Web 存储的数据存储位置取决于浏览器，并且每个浏览器的存储都是分开且独立的。



本地存储与 Cookies

初看起来，本地存储似乎与 HTTP cookies 扮演着类似的角色，但仍有一些主要区别。首先，是在服务器端读取 cookie，但是，本地存储仅可供客户端使用。如果您需要服务器端代码以根据一些存储值做出不同的响应，就应使用 cookies。但是，cookies 是与每一个 HTTP 请求一起发送到您的服务器，这可能会在带宽方面产生很大的开销。从另一方面讲，本地存储仅存储在用户的硬盘上来等待读取，所以它没有任何成本。

另外，我们可以使用本地存储来存储更大的东西。使用 cookies，我们总共只能存储 4KB 的信息，而使用本地存储，最多可存 5MB。

10.3.2 Web 存储数据的外观

存储在 Web 存储的数据以键/值对形式存储。

一些简单的键/值对的示例：

- key: name, value: Alexis;
- key: painter, value: Picasso;
- key: email, value: info@me.com。

10.3.3 获取和设置数据

与 Web 存储最相关的方法在名为 `Storage` 的对象中定义。以下是关于 `Storage` 的完整定义¹:

```
interface Storage {
    readonly attribute unsigned long length;
    DOMString key(in unsigned long index);
    getter any getItem(in DOMString key);
    setter creator void setItem(in DOMString key, in any value);
    deleter void removeItem(in DOMString key);
    void clear();
};
```

我们首先要讨论的方法是 `getItem` 和 `setItem`。我们通过调用 `setItem` 将键/值对储存在会话存储或本地存储中，并且通过调用 `getItem` 从键中获取值。

如果我们想将数据存储在会话存储中，或者从会话存储中检索数据，只需调用全局对象 `sessionStorage` 的 `setItem` 或 `getItem` 即可。如果我们想使用本地存储，可以调用全局对象 `localStorage` 的 `setItem` 或 `getItem`。在接下来的示例中，我们将条目存储在本地存储中。

当我们使用 `setItem` 方法时，必须指定保存值的键以及值本身。例如，我们要将值“6”存在键“size”中，就将调用 `setItem`:

```
localStorage.setItem("size", "6");
```

要检索存储到“size”键中的值，我们应使用 `getItem` 方法，仅指定键:

```
var size = localStorage.getItem("size");
```

10.3.4 转换存储的数据

Web 存储将所有值都存为字符串，所以如果您需要将它们作为其他类型使用时，比如数字或是对象，则需要转换它们。要将它们从字符串转换为数值，可以使用 JavaScript 的 `parseInt` 方法。

以鞋的号码为例，在 `size` 变量中所返回和储存的值将是字符串“6”，而不是

¹ <http://dev.w3.org/html5/webstorage/#the-storage-interface>

数字 6。要将它转换为数字，将使用 `parseInt`：

```
var size = parseInt(localStorage.getItem("size"));
```

10.3.5 快捷方式

我们可以继续使用 `getItem(key)` 和 `setItem(key, value)`；但是，我们也可以使用快捷方式存储和检索数据。

不使用 `localStorage.getItem(key)`，我们可以使用 `localStorage[key]`。例如，我们可以重新编写关于检索鞋号码的代码：

```
var size = localStorage["size"];
```

不使用 `localStorage.setItem(key, value)`，可以使用 `localStorage[key]=value`：

```
localStorage["size"] = 6;
```



没有键的情况！

如果您在一个没有存储的键中请求 `getItem`，会出现什么情况呢？在这种情况下，`getItem` 将返回 `null`。

10.3.6 删除条目和清除数据

要从 Web 存储中删除特定的条目，可以使用 `removeItem` 的方法。我们将想要删除的键传递给它，它将删除键及其值。

要删除用户计算机上存储的我们站点的所有值，可以使用 `clear` 方法。这将会删除为我们的域存储的所有键和值。

10.3.7 存储限制

Internet Explorer “允许 Web 应用程序存储近 10MB 的用户数据”。¹Chrome、Safari、Firefox 和 Opera 都允许存储高达 5MB 的用户数据，这是在 W3C 规范中所

¹ <http://msdn.microsoft.com/en-us/library/cc197062%28VS.85%29.aspx>

建议的数量。这个数据可能已经随着时间的推移而发生了变化，正如规范所述：“建议的限制为 5MB。欢迎实践后的反馈，并用于在将来对此建议进行更新。”另外，Opera 允许用户设置分配给 Web 存储的磁盘空间。

无需担心每个浏览器的存储量，一个更好的方法是：在存储重要数据前，测试是否超出了分配额。测试的方法是捕获 `QUOTA_EXCEEDED_ERR` 异常。以下是此方法的一个示例：

```
try
{
    sessionStorage["name"] = "Tabatha";
}
catch (exception)
{
    if (exception == QUOTA_EXCEEDED_ERR)
    {
        // we should tell the user their quota has been exceeded.
    }
}
```



Try/Catch 和异常

有时，问题发生在我们的代码中。API 的设计人员知道这一点，并且为了减轻问题的影响，他们依赖于使用异常。当意想不到的事情发生时，便会出现异常。API 的作者可以定义在几种特定问题发生时抛出一些特定的异常。然后，使用那些 API 的开发人员可以决定如何响应指定类型的异常。

为了响应异常，我们可以封装任何我们认为有可能在 `try/catch` 代码块中抛出异常的代码。这可能是您所期待的工作方式：首先，您尝试做一些事情。如果异常失败了，您可以捕获这个异常，并尝试恢复它。

要了解关于 `try/catch` 代码块的更多内容，请参见在 Mozilla Developer Networks' JavaScript Reference 的“`try...catch`”文章¹。

10.3.8 安全考虑

Web 存储具有我们所知道的基于源的安全性。这意味着通过 Web 存储从一个指

¹ <https://developer.mozilla.org/en/JavaScript/Reference/Statements/try...catch>

定域存储的数据仅能够从这个域的页面进行访问。不可能访问任何由不同域存储在 Web 存储的数据。例如，假设我们控制 `html5isgreat.com` 这个域，并且使用本地存储存储在那个站点创建的数据。而另一个域，比如说，`google.com`，便不能访问任何由 `html5isgreat.com` 存储的数据。同样，`html5isgreat.com` 也不能访问任何由 `google.com` 存储在本地存储的数据。

10.3.9 将 Web 存储添加到 The HTML5 Herald

我们可以使用 Web 存储在注册页面添加一个“在此计算机上保留我的信息”复选框。这样，一旦用户注册，那么以后在此网站上填写其他表单时，就可自动获取此信息。

让我们定义一个函数来抓取表单 `input` 元素的姓名和电子邮件地址值，再次使用 jQuery：

js/rememberMe.js (excerpt)

```
function saveData() {
    var email = $("#email").val();
    var name = $("#name").val();
}
```

我们分别在名为 `email` 和 `name` 的变量中简单地存储了电子邮件和姓名这两个表单字段栏的值。

完成检索这两个 `input` 元素中的值后，下一步是将这些值存储在 `local Storage` 中：

js/rememberMe.js (excerpt)

```
function saveData() {
    var email = $("#email").val();
    var name = $("#name").val();

    localStorage["name"] = name;
    localStorage["email"] = email;
}
```

我们也可以通过选中复选框“保存我的信息”将信息存储在本地存储中：

js/rememberMe.js (excerpt)

```
function saveData() {
    var email = $("#email").val();
    var name = $("#name").val();

    localStorage["name"] = name;
    localStorage["email"] = email;
    localStorage["remember"] = "true";
}
```

现在我们已经有了一个函数来存储访问者的姓名和电子邮件地址，如果访问者选中了复选框“在此计算机上保留我的信息”，就将调用此函数。通过这样做，我们可以观察复选框的 `change` 事件，只要复选框的状态发生改变，无论是单击它还是按下键盘，都将触发此事件：

js/rememberMe.js (excerpt)

```
$('document').ready(function() {
    $('#rememberme').change(saveData);
});
```

接下来，由于在复选框没有被选中的情况下也会触发 `change` 事件，因此让我们确认复选框是否确实被选中：

js/rememberMe.js (excerpt)

```
function saveData() {
    if ($("#rememberme").attr("checked"))
    {
        var email = $("#address").val();
        var name = $("#register-name").val();

        localStorage["name"] = name;
        localStorage["email"] = email;
        localStorage["remember"] = "true";
    }
}
```

新的一行代码调用了 jQuery 的方法 `attr("checked")`，如果复选框被选中，那么它返回的值是 `true`，否则是 `false`。

最后，我们要确保 Web 存储呈现在访问者的浏览器中：

js/rememberMe.js (excerpt)

```
function saveData() {
    if (Modernizr.localstorage) {
        if ($("#rememberme").attr("checked"))
        {
            var email = $("#address").val();
            var name = $("#register-name").val();

            localStorage["name"] = name;
            localStorage["email"] = email;
            localStorage["remember"] = "true";
        }
    }
    else
    {
        // no support for Web Storage
    }
}
```

现在，只要支持本地存储，无论何时选中复选框，我们都将存储访问者的姓名和电子邮件。问题是，我们还没有对这个数据进行任何操作！

让我们添加另一个函数来查看是否存储了姓名和电子邮件，如果已经存储，在恰当的 input 元素中填写此信息。让我们也预查一下“保存我的信息”复选框，看看我们是否已将本地存储中的“remember”键设置为“true”：

js/rememberMe.js (excerpt)

```
function loadStoredDetails() {
    var name = localStorage["name"];
    var email = localStorage["email"];
    var remember = localStorage["remember"];

    if (name) {
        $("#name").val(name);
    }
    if (email) {
        $("#email").val(email);
    }
    if (remember == "true") {
        $("#rememberme").attr("checked", "checked");
    }
}
```

我们要在执行这些操作之前，再一次检查并确认浏览器是否支持 Web 存储：

js/rememberMe.js (excerpt)

```
function loadStoredDetails() {
    if (Modernizr.localstorage) {
        var name = localStorage["name"];
        var email = localStorage["email"];
        var remember = localStorage["remember"];

        if (name) {
            $("#name").val(name);
        }
        if (email) {
            $("#email").val(email);
        }
        if (remember == "true") {
            $("#rememberme").attr("checked", "checked");
        }
    } else {
        // no support for Web Storage
    }
}
```

作为最后一步，我们将在加载页面时调用 loadStoredDetails 函数：

js/rememberMe.js (excerpt)

```
$('document').ready(function(){
    loadStoredDetails();
    $('#rememberme').change(saveData);
});
```

现在，如果用户以前访问过此页面，并选中“请在此计算机上保留我的信息”，那么他们的姓名和电子邮件就会被填充到他们接下来所访问的页面。

10.3.10 用网页审查工具查看 Web 存储值

我们可以使用 Safari 或 Chrome 的网页审查工具查看或更改本地存储中的值。在 Safari 中，我们可以查看存储在 **Storage** 选项卡下的数据，如图 10.6 所示。

在 Chrome 浏览器中，可以通过 Resources 选项卡查看数据。

由于用户拥有存储在他们硬盘中的任何数据，因此他们确实可以更改存储在

Web 存储中的数据（如果他们选择这样做）。

让我们自己测试一下。在浏览 `register.html` 页面时，您在网页审查工具的 Storage 选项卡中双击“email”值，就能够更改存储在那里的值，如图 10.7 所示。

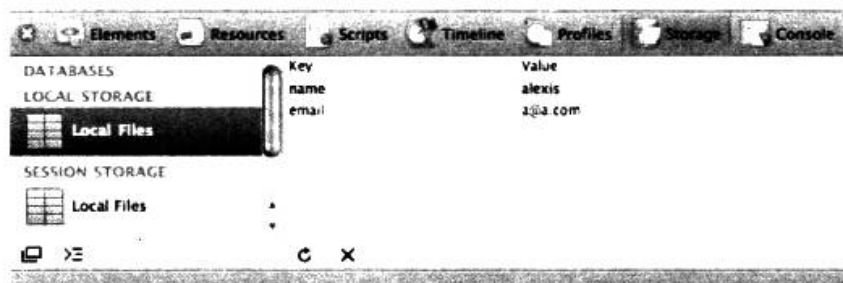


图 10.6 查看存储在本地存储和会话存储中的值

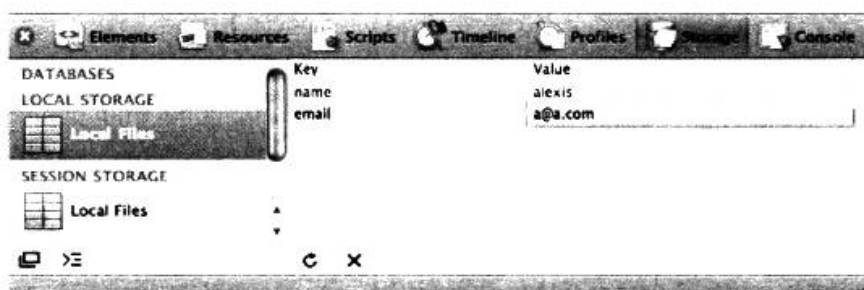


图 10.7 更改存储在 Web 存储中的值

用户拥有在自己计算机上的所有数据，作为开发人员，您不能防止用户对此进行更改。我们应该记住精明的用户有能力更改本地存储中的数据。另外，Web 存储规范规定：任何在浏览器中询问用户清除 cookies 的对话，现在也应允许他们清除本地存储。保留此信息，是由于我们不能 100% 确定我们所存储的数据是准确的，而且也不总是一直存储在那里。因此，敏感数据不应保存在本地存储中。

如果您想了解关于 Web 存储的更多信息，以下资源可供参考：

- W3C 的 Web 存储规范¹；
- Mozilla Developer Network 的 Web 存储文档²；
- IBM developerWorks 的 Web 存储教程³。

¹ <http://dev.w3.org/html5/webstorage/#the-storage-interface>

² <https://developer.mozilla.org/en/DOM/Storage>

³ <http://www.ibm.com/developerworks/xml/library/x-html5mobile2/>

10.4 其他 HTML5 API

本书还有许多其他 API 未涉及，但是，我们想在这里简单地提一下，以便您大概了解它们，此外还为您提供了一些了解更多信息的资源。

10.4.1 网络工作者

新的网络工作者 API 允许我们在背景中运行大型脚本，而不会中断主要页面或 Web 应用程序。在网络工作者出现之前，无法同时运行多个 JavaScript 脚本。您是否曾看到类似图 10.8 所示的对话框？

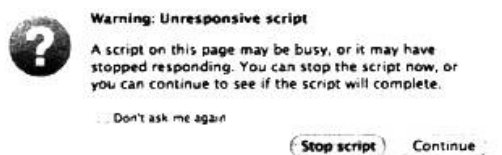


图 10.8 脚本运行时间太长以至于冻结了整个页面

有了网络工作者，我们会很少看到此类警告。新 API 允许我们执行要长时间运行的脚本，而不需要用户干预，并且我们还可以在后台运行它们的同时运行其他需要用户干预的任何脚本。此理念在编程中称为**线程**，网络工作者为我们提供了类似线程的功能。每个“工作者”处理自己的脚本块，而不会影响其他工作者或页面的其他部分。为了确保工作者之间彼此保持同步，该 API 定义了工作者之间彼此传递信息的方式。

下列浏览器支持网络工作者：

- 🍏 Safari 4 及更高版本；
- 🦊 Chrome 5 及更高版本；
- 🦖 Firefox 3.5 及更高版本；
- 🍉 Opera 10.6 及更高版本。

目前，IE、iOS 和 Android 的所有版本都不支持网络工作者。

要了解关于网络工作者的更多信息，请参见：

- 📖 HTML5 Rocks, “The Basics of Web Workers”¹；

¹ <http://www.html5rocks.com/tutorials/workers/basics/>

■ Mozilla Developer Network, “Using Web Workers”¹;

■ W3C 网络工作者规范²。

10.4.2 网络套接字

网络套接字定义“使用远程主机进行双向通信的协议”。³我们不会介绍该主题，主要有以下几个原因。第一，该 API 主要针对的是服务器端开发人员，而不是前端开发人员和设计人员。第二，网络套接字仍处于开发中，并且实际上已出现了一些安全问题。由于这些问题，Firefox 4 和 Opera 11 已默认禁用了网络套接字。⁴

下列浏览器支持网络套接字：

■ Safari 5 及更高版本；

■ Chrome 4 及更高版本；

■ Firefox 4 及更高版本（但默认是禁用的）；

■ Opera 11 及更高版本（但默认是禁用的）；

■ iOS (Mobile Safari) 4.2 及更高版本。

目前，Internet Explorer 和 Android 的所有版本都不支持网络套接字。

要了解关于网络套接字的更多信息，请参见 W3C 网站上的规范：
<http://dev.w3.org/html5/websockets/>。

10.4.3 Web SQL 和 IndexedDB

有时，Web 存储 API 提供的 SMB 的存储和简单的键值对是远远不够的。如果您需要存储大量数据，并且数据之间的关系很复杂，那么您可能需要成熟的数据库来满足自己的存储要求。

通常服务器端的数据库是独一无二的，但是目前提出了两个数据库解决方案来满足客户端的这种需求，即 Web SQL 和数据库索引 API（简称为 IndexedDB）。Web SQL 规范不再更新，尽管目前看起来 IndexedDB 似乎发展得很快，但谁将成为未来

¹ https://developer.mozilla.org/En/Using_web_workers

² <http://dev.w3.org/html5/workers/>

³ <http://www.w3.org/TR/websockets/>

⁴ 参见 <http://hacks.mozilla.org/2010/12/websockets-disabled-in-firefox-4/> 和 <http://dev.opera.com/articles/view/introducing-web-sockets/>

浏览器在大型数据存储方面的标准仍有待观察。

下列浏览器支持 Web SQL:

- Safari 3.2 及更高版本;
- Chrome;
- Opera 10.5 及更高版本;
- iOS (Mobile Safari) 3.2 及更高版本;
- Android 2.1 及更高版本。

目前, Internet Explorer 和 Firefox 的所有版本都不支持 Web SQL, 只有 Firefox 4 支持 IndexedDB。

如果您想要了解更多信息, 下面是一些很好的资源:

- Mark Pilgrim 关于 HTML5 本地存储的概述¹;
- W3C 的 IndexedDB 规范²;
- W3C 的 Web SQL 规范³。

10.5 返回到绘制面板

在本章中, 我们简单地介绍了最新一代浏览器中可用的新 JavaScript API。尽管并不是所有浏览器都支持这些新 API, 但是类似 Modernizr 这样的工具可以帮助我们逐渐将它们纳入实际项目应用中, 为我们的网站和应用程序提供另一种维度。

在下一章, 也就是最后一章, 我们将介绍另一个 API, 以及两种在浏览器中创建复杂图形的技术。这些为创建更加生动的 Web 应用程序提供了更多的潜在途径。

¹ <http://diveintohtml5.org/storage.html#future>

² <http://dev.w3.org/2006/webapi/IndexedDB/>

³ <http://dev.w3.org/html5/webdatabase/>

Chapter 11

画布、SVG 和拖放

The HTML5 Herald 已经成为了一个十分动态的“旧式”报纸！我们已经用新的 video 元素添加了视频，使我们的网站可以脱机使用，并添加了技术支持来记住用户的名字及电子邮件地址，此外还使用地理定位服务来检测用户的位置。

但是我们还可以做一些事情，使它变得更有趣。首先，由于颜色问题，视频与报纸的其余部分还有些不一致。其次是地理定位功能，当用户快速移动时，它可以使用进度指示器让用户知道我们没有丢下它们置之不理。最后，再为我们的页面添加一个动态块就更加完美了。我们将使用在本章中讨论的 API 处理这 3 个功能：画布、SVG 和拖放。

11.1 画布

使用 HTML5 画布的 API，我们便不再局限于只能在网站上绘制矩形。我们可以通过 JavaScript 绘制想要的任何东西。这样，就通过避免从网络上下载图像而提高了网站的性能。使用画布，我们可以绘制形状和线条、弧线和文字、渐变和图案。另外，画布为我们提供了调控图像甚至视频像素的功能。我们首先介绍一些画布的基本绘制功能，然后继续使用其功能改善我们的视频——用现代的颜色修饰我们的视频，并将它转化为传统的黑色和白色，使之与 The HTML5 Herald 的整体外观相匹配。

下列浏览器支持画布 2D 内容规范：

- Safari 2.0 及更高版本;
- Chrome 3.0 及更高版本;
- Firefox 3.0 及更高版本;
- Internet Explorer 9.0 及更高版本;
- Opera 10.0 及更高版本;
- iOS (Mobile Safari) 1.0 及更高版本;
- Android 1.0 及更高版本。

11.1.1 关于画布的一些历史

画布最初由 Apple 开发。由于 Apple 已经使用 Quartz 2D 框架，用于二维空间的绘制，因此它们在技术上具有领先性，并且以该 Quartz 2D 框架为基础，创建了 HTML5 画布的很多概念。然后，画布被 Mozilla 和 Opera 采用，并由 WHATWG 对其进行标准化（随后，W3C 接手该部分和 HTML5 的其余部分）。

这里还有一些好消息。如果您渴望进行 iPhone、iPad（简称为 iOS）或 Mac 的开发，则这里所学到的画布知识将有助于您理解 Quartz 2D 的一些基本概念。如果您已经进行过 Mac 或 iOS 开发，并且使用过 Quartz 2D，那么您应该对画布的一些概念十分熟悉。

11.1.2 创建画布元素

使用画布的第一步是在页面添加一个 canvas 元素。

`canvas/demo1.html (excerpt)`

```
<canvas>
Sorry! Your browser doesn't support Canvas.
</canvas>
```

canvas 标签之间的文字仅仅在用户浏览器不支持 canvas 元素时出现。

由于画布上的绘制是使用 JavaScript 来实现的，因此我们需要一种方式从 DOM 抓取元素。我们将通过为 canvas 指定一个 id 来达到此目的。

canvas/demo1.html (excerpt)

```
<canvas id="myCanvas">
Sorry! Your browser doesn't support Canvas.
</canvas>
```

在画布上的所有绘制都要通过 JavaScript 进行，所以确保在我们的网页准备好时，调用 JavaScript 函数。我们将在页面底部的 script 元素添加 jQuery 的文档就绪检查。

canvas/demo1.html (excerpt)

```
<script>
$('document').ready(function(){
    draw();
});
</script>
```

canvas 元素有 width 和 height 属性，也应将它们设置为：

canvas/demo1.html (excerpt)

```
<canvas id="myCanvas" width="200" height="200">
Sorry! Your browser doesn't support Canvas.
</canvas>
```

最后，让我们使用 CSS 为画布添加一个边框，能够在视觉上分辨它。画布没有默认样式，所以除非为它指定一种边框，否则在页面上很难看到它的位置：

css/canvas.css (excerpt)

```
#myCanvas {
    border: dotted 2px black;
}
```

现在我们已经设计了它的样式，可以在页面上看到 canvas 容器了，如图 11.1 所示。



图 11.1 一个虚线边框的空白画布

11.1.3 在画布上绘制

在画布上的所有绘制都要通过画布 JavaScript API 实现。在页面就绪的时候，我们调用一个名为 `draw()` 的函数，让我们继续并创建这个函数。我们将在 `script` 元素上添加此函数。第一步是在我们的页面上抓取 `canvas` 元素：

canvas/demo1.html (excerpt)

```
<script>
:
function draw() {
    var canvas = document.getElementById("myCanvas");
}
</script>
```

11.1.4 获取背景

当将 `canvas` 元素存储在一个变量中以后，我们便需要设置画布的背景。背景是呈现绘制的地方。目前，只能广泛支持二维背景下的绘制。W3C 画布规范在 `CanvasRenderingContext2D` 对象中定义了背景。我们在画布上用于绘制的多种方法就是此对象的方法。

我们通过调用 `getContext` 方法获取绘制背景，由于我们将使用二维绘制，因此将字符串“2d”作为参数传递给此方法：

canvas/demo1.html (excerpt)

```
function draw() {
    var canvas = document.getElementById("myCanvas");
    var context = canvas.getContext("2d");
}
```

由 `getContext` 返回的对象是 `CanvasRenderingContext2D` 对象。在本章中，为简单起见，我们将它简称为“背景对象”。



WebGL

WebGL 是一个由 Khronos Group 以及 WebGL 工作小组管理的新的 3D 图形 API。WebGL 工作小组包括 Apple、Mozilla、Google 和 Opera。

通过将 WebGL 和 HTML5 画布结合起来,您便可以使用 3D 绘制。目前,Firefox 4 及更高版本、Chrome 8 及更高版本和 Safari 6 都支持 WebGL。如果想了解更多相关信息,请参见 <http://www.khronos.org/webgl/>。

11.1.5 用颜色填充画笔

在开始之前,首先您必须在常规绘制画布上用颜料填充画笔。在 HTML5 画布中,您也必须做同样的事情,我们所采用的属性是 `strokeStyle` 或 `fillStyle`。在背景对象上设置这两个属性。并且两个属性都可以将下面 3 个值之一设置为它的值:代表一种颜色的字符串、`CanvasGradient` 或 `CanvasPattern`。

让我们开始使用颜色字符串来设计描边样式。您可以将描边认为是您将要绘制形状的边框。要绘制一个带有红色边框的矩形,首先定义描边的颜色:

canvas/demo1.html (excerpt)

```
function draw() {
    var canvas = document.getElementById("myCanvas");
    var context = canvas.getContext("2d");
    context.strokeStyle = "red";
}
```

绘制一个红色边框、里面填充为蓝色的矩形,我们必须也要定义填充色:

canvas/demo1.html (excerpt)

```
function draw() {
    :
    context.fillStyle = "blue";
}
```

我们可以使用任何 CSS 颜色值设置描边或填充的颜色,只要我们将它规定为一个字符串:十六进制的值,比如 `#00FFFF`;颜色名称,比如 `red` 或 `blue`;或 RGB 值,比如 `rgb(0,0,255)`。我们甚至可以使用 `RGBA` 来设置半透明描边或填充颜色。让我们将蓝色填充更改为一个 50%透明的蓝色。

canvas/demo1.html (excerpt)

```
function draw() {
    :
    context.fillStyle = "rgba(0, 0, 255, 0.5)";
}
```

11.1.6 在画布上绘制矩形

一旦我们定义了描边和填充的颜色，我们将准备绘制！让我们从绘制矩形开始。通过调用 `fillRect` 和 `strokeRect` 方法进行绘制。这两种方法都需要在开始绘制填充色或描边，以及矩形宽度和高度的地方，使用 X 和 Y 坐标。我们将从距离顶部 10 像素及距离画布左上角左边起 10 像素的地方开始添加描边和填充色。

[canvas/demo1.html \(excerpt\)](#)

```
function draw() {
  :
  context.fillRect(10,10,100,100);
  context.strokeRect(10,10,100,100);
}
```

这样将创建了一个半透明蓝色填充及红色边框的矩形，如图 11.2 所示。



图 11.2 一个简单的矩形——第一次在画布上绘制，已经很不错了

11.1.7 画布坐标系统

正如您所猜测的，在 `canvas` 元素中的坐标系统不同于您在数学课上学到的 Cartesian 坐标系统。在画布坐标系统中，左上角的坐标是 (0,0)。如果画布是 200 像素×200 像素的，那么右下角的坐标是 (200,200)，如图 11.3 所示。

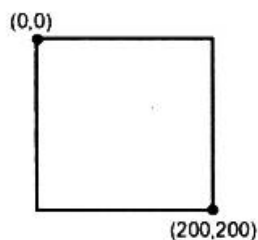


图 11.3 画布坐标系统是从上到下，从左到右

11.1.8 fillStyle 的变化

除了颜色作为我们的 `fillStyle`，我们还可以使用 `CanvasGradient` 或 `CanvasPattern`。

我们通过调用 `createPattern` 方法创建 `CanvasPattern`。`CreatePattern` 需要两个参数：创建图案的图形，如何重复使用图形。重复值是一个字符串，有效值同在 CSS 中的值一样：`repeat`、`repeat-x`、`repeat-y` 和 `no-repeat`。

为了替代半透明蓝色的 `fillStyle`，让我们用自行车图像创建一个图案。首先，我们必须创建一个 `Image` 对象，并将 `src` 属性设置为我们的图像：

`canvas/demo2.html (excerpt)`

```
function draw() {
  :
  var img = new Image();
  img.src = "../images/bg-bike.png";
}
```

通过设置 `src` 的属性，将通知浏览器开始下载图像——如果马上尝试创建渐变色，由于仍正在加载图像，我们将会碰到一些问题。所以当浏览器完全下载图像后，我们将使用图像的 `onload` 属性创建我们的图案。

`canvas/demo2.html (excerpt)`

```
function draw() {
  :
  var img = new Image();
  img.src = "../images/bg-bike.png";

  img.onload = function() {

  };
}
```

在我们的 `onload` 事件处理程序中，我们调用 `createPattern`，将 `Image` 对象和字符串 `repeat` 传递给它，这样使我们的图像沿着 X 和 Y 轴重复。我们将 `createPattern` 的结果存储在变量 `pattern` 中，并将 `fillStyle` 设置为该变量：

canvas/demo2.html (excerpt)

```
function draw() {
    :
    var img = new Image();
    img.src = "../images/bg-bike.png";
    img.onload = function() {
        pattern = context.createPattern(img, "repeat");
        context.fillStyle = pattern;
        context.fillRect(10,10,100,100);
        context.strokeRect(10,10,100,100);
    };
}
```



匿名函数

您可能会问自己,“在调用 `img.onload` 前的 `function` 语句是什么?”这是一个匿名函数。匿名函数,正如您所猜测的,除了没有名字,都与常规函数非常相像。

当您在事件侦听器中看到一个匿名函数时,便意味着那个匿名函数被绑定到该事件。换句话说,在触发 `load` 事件时,将运行匿名函数中的代码。

现在,矩形的填充是一个由自行车图像组成的图案,如图 11.4 所示。

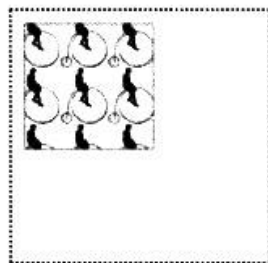


图 11.4 画布上的填充图案

我们也可以创建一个 `CanvasGradient` 作为 `fillStyle` 使用。要创建一个 `CanvasGradient`,我们可以调用两个方法中的一个: `createLinearGradient (x0, y0, x1, y1)` 或 `createRadialGradient (x0, y0, r0, x1, y1, r1)`; 然后我们为渐变再添加一个或多个色标。

`createLinearGradient` 的 `x0` 和 `y0` 表示渐变的开始位置。`x1` 和 `y1` 表示结束位置。

要创建一个开始于画布顶端、颜色逐渐混合直到底部的颜色渐变,我们首先需要定义原点 (0,0),以及结束点 (0,200):

canvas/demo3.html (excerpt)

```
function draw() {
    :
    var gradient = context.createLinearGradient(0, 0, 0, 200);
}
```

接下来，我们规定色标。色标的方法是 `addColorStop(offset, color)`。

`offset` 是一个 0 到 1 之间的值。一个为 0 的 `offset` 是在渐变的开始端，而为 1 的 `offset` 是在另一端。`color` 是一个字符串值，由于与 `fillStyle` 一起使用，因此它可以是颜色名字、十六位进制颜色值、`rgb()` 值或 `rgba()` 值。

使渐变开始为蓝色，并在一半处与白色混合并开始渐变，我们可以规定蓝色标的 `offset` 值为 0，紫色标的 `offset` 值为 1：

canvas/demo3.html (excerpt)

```
function draw() {
  :
  var gradient = context.createLinearGradient(0, 0, 0, 200);
  gradient.addColorStop(0, "blue");
  gradient.addColorStop(1, "white");
  context.fillStyle = gradient;
  context.fillRect(10, 10, 100, 100);
  context.strokeRect(10, 10, 100, 100);
}
```

图 11.5 是将 `CanvasGradient` 设置为矩形 `fillStyle` 的结果。

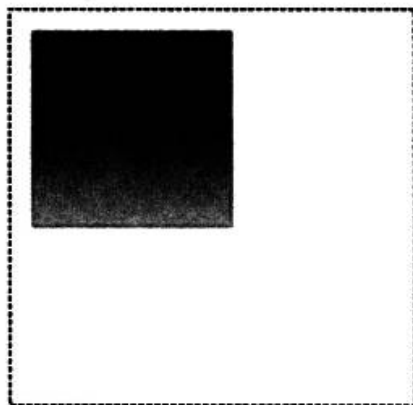


图 11.5 用画布创建线性渐变

11.1.9 通过创建路径绘制其他形状

我们不仅限于绘制矩形，也可以绘制任何想要的图形！与矩形和正方形不同，没有内置方法绘制圆形或其他形状。要绘制更多的有趣图形，我们必须首先设置形状的路径。

路径为线条、弧线以及形状提供了设计图纸，但是只有描边以后，路径才是

可见的！当我们绘制矩形的时候，我们首先设置 `strokeStyle`，然后调用 `fillRect`。对于更复杂的形状，我们需要采取 3 个步骤：设置路径，给路径描边，并填充路径。由于是绘制矩形，因此我们可以仅给路径描边，或填充路径，或者我们两者都做。

让我们开始绘制一个简单的圆形：

canvas/demo4.html (excerpt)

```
function draw() {
  var canvas = document.getElementById("myCanvas");
  var context = canvas.getContext("2d");

  context.beginPath();
}
```

现在我们需要创建一个弧线。弧线是圆形的一部分；但还没有创建圆形的方法，我们可以简单地绘制一个 360° 的弧线。我们使用 `arc` 方法创建它：

canvas/demo4.html (excerpt)

```
function draw() {
  var canvas = document.getElementById("myCanvas");
  var context = canvas.getContext("2d");

  context.beginPath();
  context.arc(50, 50, 30, 0, Math.PI*2, true);
}
```

`arc` 方法的参数如下：`arc(x,y,radius,startAngle,endAngle,anticlockwise)`。

`x` 和 `y` 表示弧线路径在画布的起始位置，将它设想为圆心，`radius` 是半径。

`startAngle` 和 `endAngle` 表示沿圆周的开始及结束的角度。角度的单位是弧度，那么圆周是 2π 弧度。我们想画一个完整的圆，所以我们使用 2π 为 `endAngle` 的值。在 JavaScript 中，我们可以通过 `Math.PI` 乘以 2 来得到此值。

`anticlockwise` 是一个可选的参数。如果您想绘制弧度时，采用逆时针而不是顺时针，那么请将此值设置为 `true`。由于我们要画一个整圆，从哪个方向画都无所谓，因此我们忽略这个参数。

我们现在已经画完了这个圆，下一步是关闭路径。我们采用 `closePath` 方法：

canvas/demo4.html (excerpt)

```
function draw() {
    var canvas = document.getElementById("myCanvas");
    var context = canvas.getContext("2d");

    context.beginPath();
    context.arc(100, 100, 50, 0, Math.PI*2, true);
    context.closePath();
}
```

现在我们有了路径！但是如果不进行描边或填充，我们将不能看到它。这样，如果想给它一个边框，我们就必须设置一个 `strokeStyle`，如果还要给圆填充颜色，还要设置一个 `fillStyle`。默认情况下，描边的宽度是 1 像素，它存储在 `context` 对象的 `lineWidth` 属性中。我们将 `lineWidth` 设置为 3，这样边框会稍微大一些。

canvas/demo4.html (excerpt)

```
function draw() {
    var canvas = document.getElementById("myCanvas");
    var context = canvas.getContext("2d");

    context.beginPath();
    context.arc(50, 50, 30, 0, Math.PI*2, true);
    context.closePath();
    context.strokeStyle = "red";
    context.fillStyle = "blue";
    context.lineWidth = 3;
}
```

最后，我们将对路径进行描边和填充。请注意，这一次我们使用的方法名称不同于在矩形中使用的方法名称。要填充路径，您只要简单地调用 `fill`，并通过调用 `stroke` 进行描边：

canvas/demo4.html (excerpt)

```
function draw() {
    var canvas = document.getElementById("myCanvas");
    var context = canvas.getContext("2d");

    context.beginPath();
    context.arc(100, 100, 50, 0, Math.PI*2, true);
    context.closePath();
```

```

context.strokeStyle = "red";
context.fillStyle = "blue";
context.lineWidth = 3;
context.fill();
context.stroke();
}

```

图 11.6 显示了绘制完成的圆形。

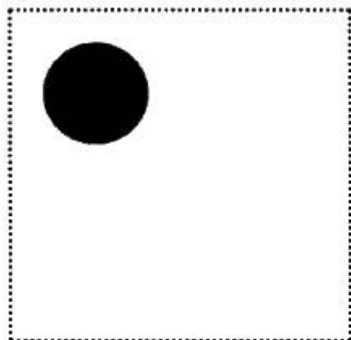


图 11.6 闪亮的新圆圈

如果您想了解关于绘制形状的更多信息，Mozilla Developer Network 提供了非常好的教程¹。

11.1.10 存储画布绘制

如果我们使用 Canvas API 以编程的方式创建一个图像，决定存储绘制的本地副本，就可以使用 API 的 `toDataURL` 方法，将绘制存为 PNG 或 JPEG 文件。

为了保存我们刚刚绘制的圆，我们可以在 HTML 上添加一个新按钮，当单击按钮时，作为图像在新窗口打开画布绘制。为此，让我们先定义一个新的 JavaScript 函数：

```

function saveDrawing() {
    var canvas = document.getElementById("myCanvas");
    window.open(canvas.toDataURL("image/png"));
}

```

canvas/demo5.html (excerpt)

接下来，我们在 HTML 上添加一个按钮，并在单击它时调用函数：

¹ https://developer.mozilla.org/en/Canvas_tutorial/Drawing_shapes

canvas/demo5.html (excerpt)

```

<canvas id="myCanvas" width="200" height="200">
Sorry! Your browser doesn't support Canvas.
</canvas>
<form>
  <input type="button" name="saveButton" id="saveButton"
  value="Save Drawing">
</form>
:
<script>

$('document').ready(function(){
  draw();
  $('#saveButton').click(saveDrawing);
});
:

```

当单击按钮时，一个新的窗口或标签打开，并加载一个 PNG 文件，如图 11.7 所示。

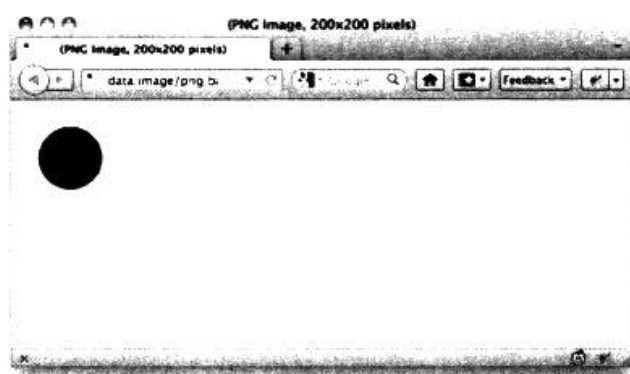


图 11.7 在新窗口中加载的图像

要了解关于将画布绘制存储为文件的更多信息，请参见 W3C 画布规范¹和 Mozilla 的画布代码片段的 Saving a canvas image to file 部分²。

11.1.11 在画布上绘制图像

我们也可以在 canvas 元素中绘制图像。在本例中，我们将在画布中再绘制一个

¹ <http://www.w3.org/TR/html5/the-canvas-element.html#dom-canvas-todataurl>

² https://developer.mozilla.org/en/Code_snippets/Canvas

已在页面存在的图像。

为了说明，我们将使用 HTML5 的图标¹作为我们下面几个示例中的图像。让我们将它放在 `img` 元素中，这样就将它添加到了页面中。

canvas/demo6.html (excerpt)

```
<canvas id="myCanvas" width="200" height="200">
Your browser does not support canvas.
</canvas>

```

下一步，在抓取 `canvas` 元素以及建立画布背景后，我们可以通过 `document.getElementById` 从页面抓取图像：

canvas/demo6.html (excerpt)

```
function draw() {
    var canvas = document.getElementById("myCanvas");
    var context = canvas.getContext("2d");
    var image = document.getElementById("myImageElem");
}
```

我们将使用前面用过的 CSS 使 `canvas` 元素可见：

css/canvas.css (excerpt)

```
#myCanvas {
    border: dotted 2px black;
}
```

让我们稍作修改以隔开画布及图像：

css/canvas.css (excerpt)

```
#myCanvas {
    border: dotted 2px black;
    margin: 0px 20px;
}
```

图 11.8 在图像旁边显示了一个空的画布。

¹ <http://www.w3.org/html/logo/>

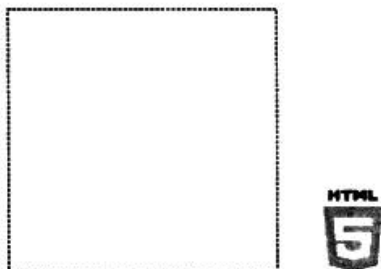


图 11.8 在页面中显示了一个图像和一个空的画布，还没有做很多工作

我们可以使用画布的 `drawImage` 方法重画页面的图像，并将它画到画布中：

canvas/demo6.html (excerpt)

```
function draw() {
  var canvas = document.getElementById("myCanvas");
  var context = canvas.getContext("2d");
  var image = document.getElementById("myImageElem");
  context.drawImage(image, 0, 0);
}
```

因为我们在坐标 (0,0) 处绘制了图像，图像在画布的左上方显示，如图 11.9 所示。

我们可以通过更改传给 `drawImage` 的 X 和 Y 坐标，改为在画布的中心绘制图像。由于图像是 64 像素×64 像素，画布是 200 像素×200 像素，如果我们在坐标 (68, 68)¹，图像将在画布的中央，如图 11.10 所示。

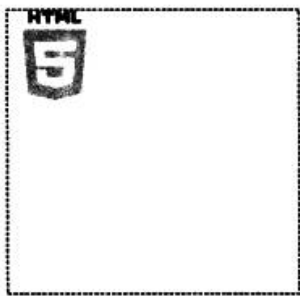


图 11.9 在画布上重新绘制了图像

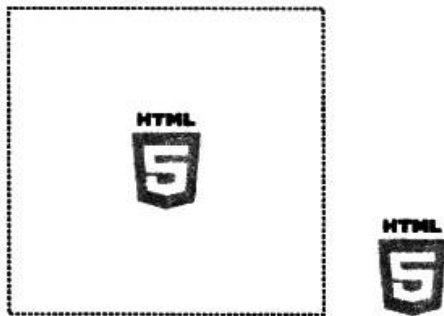


图 11.10 在画布上重新绘制了图像

11.1.12 处理图像

从页面重画一个图像元素到画布上是相当平淡无奇的。确实与使用 `img` 元素没

¹ 画布的一半大小减去图像的一半大小：(200/2) - (64/2) = 68

有任何区别！而真正有趣的是：在画布上完成绘制以后如何处理图像。

一旦在画布上完成了绘制图像，我们就可以使用 Canvas API 的 `getImageData` 方法。例如，想将商标颜色转换成黑白色，我们可以使用 Canvas API 中的方法。

`getImageData` 将返回一个 `ImageData` 对象，`ImageData` 对象包含 3 个属性：`width`、`height` 和 `data`。前两个不言自明，但是最后一个，`data`，确实非常有趣。

`data` 以数组的形式，包含关于在 `ImageData` 对象的像素信息。在画布上的每一个像素在 `data` 数组中有 4 个值——分别对应像素的 R、G、B 和 A 值。

`getImageData` 方法使我们能够检测画布中的一小部分，所以让我们使用这个功能熟悉一下数据数组。`getImageData` 需要 4 个参数，对应的是我们要检测画布的矩形画布的 4 个角。如果我们在画布的一小部分中调用 `getImageData`，指明 `context`。`getImageData(0,0,1,1)`，那么我们只检测一个像素（从 (0,0) 到 (1,1) 的矩形）。关于这个单独像素所返回的数组包含 4 项，分别是红色、绿色、蓝色和 `alpha` 值：

```
function draw() {
    var canvas = document.getElementById("myCanvas");
    var context = canvas.getContext("2d");
    var image = document.getElementById("myImageElem");
    // draw the image at x=0 and y=0 on the canvas
    context.drawImage(image, 68, 68);
    var imageData = context.getImageData(0, 0, 1, 1);
    var pixelData = imageData.data;
    alert(pixelData.length);
}
```

警告提示确认画布一个像素的数据数组将有 4 个值，如图 11.11 所示。

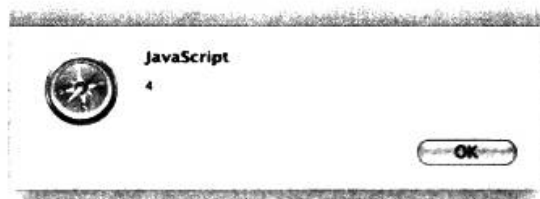


图 11.11 一个单独像素的数据数组包含 4 个值

11.1.13 将彩色图像转换为黑白图像

让我们来看看如何在画布上使用 `getImageData` 将全彩色图像转换为黑白图

像。假设我们已经在画布上放置了一个图像，像上面那样，我们可以使用 `for` 循环迭代通过图像的每个像素，并改变它的灰度。

首先，我们将调用 `getImageData(0,0,200,200)` 检索整个画布。然后，需要抓取每个像素的红色、绿色和蓝色，它们在数组中以这样的顺序排列：

canvas/demo7.html (excerpt)

```
function draw() {
    var canvas = document.getElementById("myCanvas");
    var context = canvas.getContext("2d");
    var image = document.getElementById("myImageElem");
    context.drawImage(image, 68, 68);

    var imageData = context.getImageData(0, 0, 200, 200);
    var pixelData = imageData.data;

    for (var i = 0; i < pixelData.length; i += 4) {
        var red = pixelData[i];
        var green = pixelData[i + 1];
        var blue = pixelData[i + 2];
    }
}
```

注意，`for` 循环语句每一次的增量是 4，而不是通常的 1。这是因为每一个像素在 `imageData` 数组中取 4 个值——每一个数字分别代表 R、G、B 和 A 值。

接下来，我们必须确定当前像素的灰度值。事实证明，使用一个数学公式将 RGB 转换为灰度，您只需要简单地将每一个红色、绿色和蓝色值乘以特定的数字，请看下面的代码：

canvas/demo7.html (excerpt)

```
:
for (var i = 0; i < pixelData.length; i += 4) {
    var red = pixelData[i];
    var green = pixelData[i + 1];
    var blue = pixelData[i + 2];

    var grayscale = red * 0.3 + green * 0.59 + blue * 0.11;
}
:
```

现在，我们已经有了一个正确的灰度值，我们将这些红色、绿色和蓝色值重新存储到 `data` 数组中：

canvas/demo7.html (excerpt)

```

:
for (var i = 0; i < pixelData.length; i += 4) {
    var red = pixelData[i];
    var green = pixelData[i + 1];
    var blue = pixelData[i + 2];

    var grayscale = red * 0.3 + green * 0.59 + blue * 0.11;

    pixelData[i] = grayscale;
    pixelData[i + 1] = grayscale;
    pixelData[i + 2] = grayscale;
}
:

```

所以，现在已经通过分别转换每个像素为灰度的方式，更改了像素数据。那么最后一步呢？将我们所更改的图像通过 `putImageData` 方法放回到画布中。这个方法能够达到我们预期的目的：它读取画布数据并将其写到画布上。下面是实际运行该方法的代码：

canvas/demo7.html (excerpt)

```

function draw() {
    var canvas = document.getElementById("myCanvas");
    var context = canvas.getContext("2d");
    var image = document.getElementById("myImageElem");
    context.drawImage(image, 60, 60);

    var imageData = context.getImageData(0, 0, 200, 200);
    var pixelData = imageData.data;

    for (var i = 0; i < pixelData.length; i += 4) {
        var red = pixelData[i];
        var green = pixelData[i + 1];
        var blue = pixelData[i + 2];

        var grayscale = red * 0.3 + green * 0.59 + blue * 0.11;

        pixelData[i] = grayscale;
        pixelData[i + 1] = grayscale;
        pixelData[i + 2] = grayscale;
    }
    context.putImageData(imageData, 0, 0);
}

```


我们已经在画布上绘制完成了一个黑白版本的图像。

11.1.14 getImageData 的安全性错误

在 Chrome 或 Firefox 中尝试此代码，您可能会注意到此代码不能运行——画布上的图像还是彩色的。这是因为，如果您尝试在桌面上的 HTML 文件里转换图像，同时 HTML 文件也在您的桌面上，那么在这两个浏览器中的 `getImageData` 就会发生错误。这个错误是一个安全性错误，尽管它在我们的示例中是多余的。

Chrome 和 Firefox 试图阻止真正的问题是在一个域的用户操纵其他域上的图像。例如，停止我们从 <http://google.com/> 下载官方标志，然后对其像素数据进行处理。

W3C Canvas 规范¹这样形容它：

如果来自一个域的脚本可以访问另一个域（不同的域）的图像信息（如：读取像素），就会造成信息泄露。为了解决这个问题，为 canvas 元素定义了一个标志，表明它们是否是 `origin-clean`。

如果您想处理的图像与用来处理图像的 JavaScript 不在同一个域，应将 `origin-clean` 标志设置为 `false`。不幸的是，在 Chrome 和 Firefox 浏览器中，当您测试硬盘中的文件进行测试时——它们认为此文件在不同的域中，这个 `origin-clean` 标志也设置为 `false`。

如果您想使用画布测试 Firefox 或 Chrome 中的像素处理，就需要在您的计算机上运行网络服务器进行测试，或在真正的网络服务器上在线测试。

11.1.15 用画布测试视频

我们可以利用前面已经编写好的用于将彩色图像转换为黑白图像的代码，并对其改进，将我们的彩色视频转换为黑白视频，以匹配 The HTML5 Herald 页面怀旧的感觉。我们将在一个单独的、新的名为 `videoToBW.js` 的 JavaScript 文件中编写我们的代码，这样我们就可以在网站主页上包含它。

在文件的开头，我们通常设置画布及其背景：

¹ <http://dev.w3.org/html5/2dcontext/>

js/videoToBW.js (excerpt)

```
function makeVideoOldTimey ()
{
  var video = document.getElementById("video");
  var canvas = document.getElementById("canvasOverlay");
  var context = canvas.getContext("2d");
}
```

接下来，我们将添加一个新的事件侦听器，对 video 元素触发的 play 事件做出反应。

我们希望视频开始播放时调用 draw 函数。要做到这一点，我们将为对 play 事件做出响应的 video 元素添加一个事件侦听器：

js/videoToBW.js (excerpt)

```
function makeVideoOldTimey ()
{
  var video = document.getElementById("video");
  var canvas = document.getElementById("canvasOverlay");
  var context = canvas.getContext("2d");

  video.addEventListener("play", function(){
    draw(video,context,canvas);
  },false);
}
```

在触发 play 事件的时候将调用 draw 函数，并向该函数传递 video、context 和 canvas 对象等参数。我们在这里使用匿名函数，而不使用正常的命名函数，因为当将命名函数声明为事件处理器的时候，我们确实不能将参数传递给它们。

由于我们要将若干个参数传递给 draw 函数 (video、context 和 canvas)，因此我们必须从匿名函数内调用它。

让我们看一下 draw 函数：

js/videoToBW.js (excerpt)

```
function draw(video, context, canvas)
{
  if (video.paused || video.ended)
  {
    return false;
  }
}
```

```

    }

    drawOneFrame(video, context, canvas);
}

```

在做其他事情之前，我们需要检测视频是否暂停或结束，在暂停或结束的情况下，我们将通过返回 `false` 中断函数。否则，我们将继续运行 `drawOneFrame` 函数。`drawOneFrame` 函数几乎与我们上面将彩色图像转换为黑白图像的代码相同，唯一不同的是我们将 `video` 元素放在画布上，而不是静态图像上：

js/videoToBW.js (excerpt)

```

function drawOneFrame(video, context, canvas){
    // draw the video onto the canvas
    context.drawImage(video, 0, 0, canvas.width, canvas.height);

    var imageData = context.getImageData(0, 0, canvas.width,
    canvas.height);
    var pixelData = imageData.data;
    // Loop through the red, green and blue pixels,
    // turning them grayscale
    for (var i = 0; i < pixelData.length; i += 4) {
        var red = pixelData[i];
        var green = pixelData[i + 1];
        var blue = pixelData[i + 2];
        //we'll ignore the alpha value, which is in position i+3

        var grayscale = red * 0.3 + green * 0.59 + blue * 0.11;

        pixelData[i] = grayscale;
        pixelData[i + 1] = grayscale;
        pixelData[i + 2] = grayscale;
    }

    imageData.data = pixelData;

    context.putImageData(imageData, 0, 0);
}

```

我们完成一帧后，下一步做什么呢？我们需要画下一帧。`setTimeout` 方法允许我们重复地调用 `draw` 函数，而且期间没有停顿：最后参数是延迟的值——或者是在调用函数之前，浏览器要一直等待的时间，单位为毫秒。由于将

它设置为 0，因此我们基本上是连续运行 draw 函数。这种运行状况将持续到视频结束或暂停：

js/videoToBW.js (excerpt)

```
function draw(video, context, canvas) {
  if (video.paused || video.ended)
  {
    return false;
  }

  var status = drawOneFrame(video, context, canvas);

  // Start over!
  setTimeout(function(){ draw(video, context, canvas); }, 0);
}
```

最终结果是什么呢？一段飞机起飞的彩色视频现在以黑白色播放。

11.1.16 在画布上显示文字

如果从计算机的文档中看 The HTML5 Herald，在 Firefox 和 Chrome 中处理整个视频时，就像处理简单的图像一样，我们将遇到安全错误。

我们可以添加一些错误检查，即使使用本地计算机在 Chrome 或 Firefox 中观看视频，也能够使我们的视频正常播放。

第一步是添加 try/catch 代码块捕获错误：

js/videoToBW.js (excerpt)

```
function drawOneFrame(video, context, canvas){
  context.drawImage(video, 0, 0, canvas.width, canvas.height);

  try {
    var imageData = context.getImageData(0, 0, canvas.width,
    ↪ canvas.height);
    var pixelData = imageData.data;
    for (var i = 0; i < pixelData.length; i += 4) {
      var red = pixelData[i];
      var green = pixelData[i + 1];
      var blue = pixelData[i + 2];
      var grayscale = red * 0.3 + green * 0.59 + blue * 0.11;
    }
  }
}
```

```

        pixelData[i] = grayscale;
        pixelData[i + 1] = grayscale;
        pixelData[i + 2] = grayscale;
    }

    imageData.data = pixelData;
    context.putImageData(imageData, 0, 0);
}
catch (err) {
    // error handling code will go here
}
}

```

在试图调用 `getImageData` 发生错误时，将给用户某种信息，以便为它们提示有可能发生的错误。我们将使用 Canvas API 的 `fillText` 方法。

在画布上写任何文字之前，我们应该清除在画布上面绘制的所有东西。我们已经通过调用 `drawImage` 将视频的第一帧画到了画布上。那么我们如何清除它呢？

事实证明，如果我们重新设置画布的宽度和高度，就将清除画布上的所有东西。好，让我们重新设置宽度：

js/videoToBW.js (excerpt)

```

function drawOneFrame(video, context, canvas){
    context.drawImage(video, 0, 0, canvas.width, canvas.height);

    try {
        ...
    }
    catch (err) {
        canvas.width = canvas.width;
    }
}

```

下一步，由于 `canvas` 元素被放在了视频的顶部，因此让我们将背景颜色从黑色改为透明色：

js/videoToBW.js (excerpt)

```

function drawOneFrame(video, context, canvas){
    context.drawImage(video, 0, 0, canvas.width, canvas.height);

    try {
        ...
    }
}

```

```

    catch (err) {
        canvas.width = canvas.width;

        canvas.style.backgroundColor = "transparent";
    }
}

```

我们在透明画布上绘制任何文字之前，首先必须设置文字的样式——与我们早先设置路径非常类似。我们使用 `fillStyle` 和 `textAlign` 方法：

js/videoToBW.js (excerpt)

```

videoToBW.js (excerpt)
function drawOneFrame(video, context, canvas){
    context.drawImage(video, 0, 0, canvas.width, canvas.height);

    try {
        :
    }
    catch (err) {
        canvas.width = canvas.width;
        canvas.style.backgroundColor = "transparent";
        context.fillStyle = "white";
        context.textAlign = "left";
    }
}

```

我们也需要设置我们所喜欢的字体。背景对象的 `font` 属性与 CSS 的 `font` 属性工作原理相同。我们将规定字体大小为 18px 和用逗号隔开的字体列表：

js/videoToBW.js (excerpt)

```

function drawOneFrame(video, context, canvas){
    context.drawImage(video, 0, 0, canvas.width, canvas.height);

    try {
        :
    }
    catch (err) {
        canvas.width = canvas.width;
        canvas.style.backgroundColor = "transparent";
        context.fillStyle = "white";
        context.textAlign = "left";
        context.font = "18px LeagueGothic, Tahoma, Geneva, sans-serif";
    }
}

```


请注意，我们使用的是 League Gothic，用@font-face 所包含的任何字体也可以在画布上使用。最后，我们绘制文字。我们使用背景对象的 fillText 方法，用此方法获取绘制的文字及所放位置的 (x,y) 坐标。因为我们要写较长的信息，所以将它分成几个部分，分别将每一个放在画布上。

js/videoToBW.js (excerpt)

```
function drawOneFrame(video, context, canvas){
    context.drawImage(video, 0, 0, canvas.width, canvas.height);

    try {
        :
    }
    catch (err) {
        canvas.width = canvas.width;
        canvas.style.backgroundColor = "transparent";
        context.fillStyle = "white";
        context.textAlign = "left";
        context.font = "18px LeagueGothic, Tahoma, Geneva, sans-serif";
        context.fillText("There was an error rendering ", 10, 20);
        context.fillText("the video to the canvas.", 10, 40);
        context.fillText("Perhaps you are viewing this page from", 10,
➔70);
        context.fillText("a file on your computer?", 10, 90);
        context.fillText("Try viewing this page online instead.", 10,
➔130);

        return false;
    }
}
```

作为最后一步，我们返回 false。这使我们在 draw 函数中检查是否有异常发生。如果有异常，我们就将使每一个视频帧停止调用 drawOneFrame，所以我们退出 draw 函数：

js/videoToBW.js (excerpt)

```
function draw(video, context, canvas) {
    if (video.paused || video.ended)
    {
        return false;
    }

    var status = drawOneFrame(video, context, canvas);
```

```

    if (status == false)
    {
        return false;
    }
    // Start over!
    setTimeout(function(){ draw(video, context, canvas); }, 0);
}

```

11.1.17 关注可访问性

画布在当前形式下的一个主要不足是它缺乏可访问性。画布并不创建 DOM 节点，它不是一个基于文本的格式，因此实际上对像屏幕阅读器这样的工具是不可见的。例如，即使我们在最后一个示例中在画布上写入文本，那些文本基本上也就是是一些像素，因此不可访问。

HTML5 社区已经意识到了这些不足的地方，但一直还没有确定最终的解决办法，关于如何访问画布的讨论一直在进行。您可以在 W3C 的 [wiki 页面](http://www.w3.org/html/wg/wiki/AddedElementCanvas)¹ 阅读关于此方面的讨论及目前提出的解决方案的编辑刊物。

11.1.18 其他阅读材料

如果需要阅读关于画布和 Canvas API 的更多信息，这里有许多很好的资源：

- “HTML5 canvas—the basics” at Dev.Opera²;
- Safari’s HTML5 Canvas Guide³。

11.2 SVG

我们已经在第 7 章中学习了一些关于 SVG 的知识，在第 7 章中我们曾使用 SVG 文件作为 Internet Explorer 9 和旧版本的 Opera 中渐变色的备用。在本章中，我们将深入钻研 SVG，并学习如何在其他方面使用它。

¹ <http://www.w3.org/html/wg/wiki/AddedElementCanvas>

² <http://dev.opera.com/articles/view/html-5-canvas-the-basics/>

³ <http://developer.apple.com/library/safari/#documentation/AudioVideo/Conceptual/HTML-canvasguide/Introduction/Introduction.html>

首先，让我们快速复习一下：SVG 表示可缩放矢量图形。SVG 是一个特定的文件格式，可让您使用 XML 描述矢量图形。总体来说，矢量图形的主要卖点是：不同于位图图像（比如 GIF、JPEG、PNG 和 TIFF），即使您将它们放大或缩小，矢量图像都可以保持其形状。我们可以在画布上使用 SVG 做许多相同的事情，包括绘制路径、形状、文字、渐变和图案。还有一些非常有用的与 SVG 相关的开放源码工具，我们会充分利用其中的一部分，为 The HTML5 Herald 的地理定位部件添加一个旋转进度指示器。

基本的 SVG，包括在 HTML 的 `img` 元素中使用 SVG，以下浏览器对其提供支持：

- Safari 3.2 及更高版本；
- Chrome 6.0 及更高版本；
- Firefox 4.0 及更高版本；
- Internet Explorer 9.0 及更高版本；
- Opera 10.5 及更高版本。

目前，Android 的浏览器还不支持 SVG。



XML

XML 表示可扩展标记语言。就像 HTML，它是一种标记语言，意味着是一个注释文本系统。就像我们使用 HTML 标签包含内容并赋给它意义，所以 XML 标签可以用于描述文件内容。

与画布不一样，用 SVG 创建的图像可通过 DOM 使用。这样就允许使用像屏幕阅读器这样的技术去观看 SVG 对象通过它的 DOM 节点所显示的内容——它也允许您使用浏览器开发商工具检查 SVG。由于 SVG 是一个 XML 文件格式，因此它也比画布更容易访问搜索引擎。

11.2.1 在 SVG 上绘制

在 SVG 中画圆要比使用画布画圆容易一些，这是一种有争议的说法。让我们来看一下在 SVG 上如何画圆：

images/circle.svg

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 400 400">
  <circle cx="50" cy="50" r="25" fill="red"/>
</svg>
```

viewBox 属性定义了 SVG 图像的开始位置、宽度和高度。

circle 元素用 cx 和 cy，也就是圆心的 X 和 Y 坐标定义了一个圆。用 r 表示半径，fill 表示填充图案。

要查看 SVG 文件，可以在支持 SVG 的任何浏览器中简单地通过 File 菜单打开它。图 11.12 显示了所绘制的圆。

我们也可以在 SVG 中绘制矩形，并可以添加描边，就像我们在使用画布时一样。

这一次，让我们利用 SVG 是一个 XML 这一点——因此以文本为基础——文本格式，并使用 desc 标签，此标签使我们能够为所绘制的图像提供一个描述：

images/rectangle.svg (excerpt)

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 400 400">
  <desc>Drawing a rectangle</desc>
</svg>
```

下一步，我们将用大量的描述矩形的属性填入 rect 标签中。其中包括所画矩形位置的 X 和 Y 坐标、矩形的宽度和高度、填充、描边以及描边的宽度：

images/rectangle.svg

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 400 400">
  <desc>Drawing a rectangle</desc>
  <rect x="10" y="10" width="100" height="100"
    fill="blue" stroke="red" stroke-width="3" />
</svg>
```

图 11.13 显示了我们所绘制的矩形。



图 11.12 使用 SVG 所绘制的圆



图 11.13 使用 SVG 所绘制的矩形

不幸的是，通常没有这么简单。如果您想创建一个复杂的图形，代码就开始看起来有一些吓人了。图 11.14 显示了 www.openclipart.org 网站上一个相当简单的星状图形。

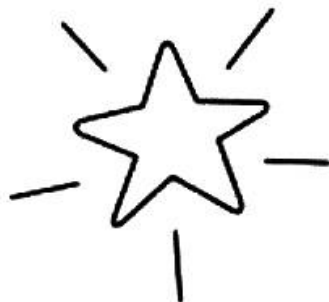


图 11.14 线描的星状图形

下面仅是绘制此 SVG 图像的前几行代码：

```
<svg xmlns="http://www.w3.org/2000/svg"
  width="122.88545" height="114.88568">
<g
  inkscape:label="Calque 1"
  inkscape:groupmode="layer"
  id="layer1"
  transform="translate(-242.42282,-449.03699)">
  <g
    transform="matrix(0.72428496,0,0,0.72428496,119.87078,183.8127)"
    id="g7153">
    <path
      style="fill:#ffffff;fill-opacity:1;stroke:#000000;stroke-width
➤:2.761343;stroke-linecap:round;stroke-linejoin:round;stroke-miterl
➤imit:4;stroke-opacity:1;stroke-dasharray:none;stroke-dashoffset:0"
      d="m 249.28667,389.00422 -9.7738,30.15957 -31.91999,7.5995 c -
➤2.74681,1.46591 -5.51239,2.92436 -1.69852,6.99979 l 30.15935,12.57
➤796 -11.80876,32.07362 c -1.56949,4.62283 -0.21957,6.36158 4.24212
➤,3.35419 l 26.59198,-24.55691 30.9576,17.75909 c 3.83318,2.65893 6
➤.12086,0.80055 5.36349,-3.57143 l -12.10702,-34.11764 22.72561,-13
➤.7066 c 2.32805,-1.03398 5.8555,-6.16054 -0.46651,-6.46042 l -33.5
➤0135,-0.66887 -11.69597,-27.26175 c -2.04282,-3.50583 -4.06602,-7.
➤22748 -7.06823,-0.1801 z"
      id="path7155"
      inkscape:connector-curvature="0"
      sodipodi:nodetypes="cccccccccccccccc" />
    
```

11.2.2 使用 Inkscape 创建 SVG 图像

为了简化一些工作（这样更理智一些），替代手动创建 SVG 图像，我们可以使用图像编辑器获取帮助。您可以使用制作 SVG 图像的一个开放源码工具 Inkscape，Inkscape 是一个输出 SVG 的开放源码矢量图形编辑器，Inkscape 可从 <http://inkscape.org/> 下载。

对于我们的旋转进度指示器，不用从头开始，我们已在公共域中找到一张很好的图像，让我们从它开始。了解公共域图像的一个很好资源是 <http://openclipart.org>，在这里可以找到无版权并免费使用的图像。图像是由一些创设者捐赠并用于公共域，即使用于商业目的，也无需请求许可。

我们将使用一个三箭头的图像作为旋转进度指示器的基底，如图 11.15 所示。原始图可在 www.openclipart.org 中找到¹。

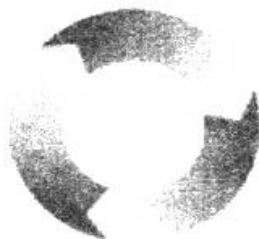


图 11.15 用于为旋转进度指示器的图像

11.2.3 SVG 过滤器

为了使我们的旋转进度指示器能够更好地与页面匹配，我们可以在 Inkscape 中使用过滤器，使其颜色变为黑白色。在 Inkscape 中打开文件，然后选择 **Filters>Color>Moonarize**。

您可能已经注意到，如果在 Safari 中测试 The HTML5 Herald，那么黑白色的旋转指示器仍为彩色。这是因为 SVG 过滤器是 SVG 的一个特殊功能，还没有在 Safari 5 中应用，但它将是 Safari 6 中的一部分。Firefox、Chrome 和 Opera 支持 SVG 过滤器。目前，所有版本的 Safari、Internet Explorer、Android 浏览器及 iOS 都不支持 SVG 过滤器。

¹ http://www.openclipart.org/people/JoBrad/arrows_3_circular_interlocking.svg

采用一个更安全的方法可以避免使用过滤器，只是简单地修改原始图的颜色即可。

我们可以在 Inkscape 中更改颜色，首先选中 `spinner.svg` 图像的 3 个箭头，然后选择 **Object>Fill and Stroke**。**Fill and Stroke** 菜单将出现在屏幕右手边，如图 11.16 所示。

从这个菜单中，我们可以通过单击 **Edit** 按钮来选择编辑目前的线性渐变。然后我们将 **Red**、**Green** 和 **Blue** 的值改为 0，这样图像就变成了黑白色。

我们已经将生成的 SVG 存为 `spinnerBW.svg`。



图 11.16 使用 Fill and Stroke 更改颜色

11.2.4 使用 Raphaël 库

Raphaël¹是一个开源的 JavaScript 库。与 SVG 一起使用，可以使绘制和动画比仅使用 SVG 容易一些。

在 Raphaël 容器中绘制图像

与使用画布一样，您也可以将图像画到用 Raphaël 创建的容器中。

让我们在主索引文件添加一个 `div`，我们将此文件作为使用 Raphaël 创建的 SVG 元素的容器。我们已经将这个 `div` 命名为 `spinner`：

```
<!-- Spinner -->
<div id="spinner">
  <img alt="Spinner" data-bbox="144 630 788 777"/>
</div>
```

css/styles.css (excerpt)

```
<article id="ad4">
  <div id="mapDiv">
    <h1 id="geoHeading">Where in the world are you?</h1>
    <form id="geoForm">
      <input type="button" id="geobutton" value="Tell us!">
    </form>
    <div id="spinner"></div>
  </div>
</article>
```

我们使用下面的 CSS，将这个 `div` 放在父 `mapDiv` 的中间：

¹ <http://dmitrybaranovskiy.github.io/raphael/>

css/styles.css (excerpt)

```
#spinner {
  position: absolute;
  top: 8px;
  left: 55px;
}
```

现在，在我们的 geolocation 文件（ipt 文件类型）中，在读取地图的时候，让我们将旋转器放在适当的位置。第一步是将 div 变为 Raphaël 容器。这就像调用 Raphaël 方法一样简单，并与宽度和高度一起，将它传递到我们使用的元素中：

js/geolocation.js (excerpt)

```
function determineLocation(){
  if (navigator.onLine) {
    if (Modernizr.geolocation) {
      navigator.geolocation.getCurrentPosition(displayOnMap);

      var container = Raphael(document.getElementById("spinner"),
        ↪125, 125);
```

接下来，我们将旋转器 SVG 图像绘制到使用 Raphaël 方法 image 新创建的容器中，它被称为 Raphaël 容器对象。此方法将路径、绘制图像的起始坐标以及图像的宽度和高度分配给图像：

js/geolocation.js (excerpt)

```
var container = Raphael(document.getElementById("spinner"),125,125);
var spinner = container.image("images/spinnerBW.svg",0,0,125,125);
```

有了这些，当我们单击地理定位小部件上的按钮时，旋转器图像便会出现。

用 Raphaël 转动旋转器

现在我们有了容器以及在里面的旋转器 SVG 图片，我们想让图像活动并使之旋转。Raphaël 有内嵌在 animate 方法中的动画功能。不过，在我们可以使用此方法前，我们需要告诉它哪个属性是用于使图像活动的。由于我们想旋转图像，因此将创建一个对象规定我们想要旋转的角度。

我们创建一个新的对象 attrToAnimate，规定我们想要转动的圈数，在这里，我们要使之转 720°（两整圈）：

js/geolocation.js (excerpt)

```
var container = Raphael(document.getElementById("spinner"),125,125);
var spinner = container.image("images/spinnerBW.png",0,0,125,125);
var attrsToAnimate = { rotation: "720" };
```

最后一步是调用 `animate` 方法，并规定动画时间的长度。在本例中，我们让它最多运行 60 秒。由于 `animate` 是以毫秒为单位的值，所以我们将传递 60000：

js/geolocation.js (excerpt)

```
var container = Raphael(document.getElementById("spinner"),125,125);
var spinner = container.image("images/spinnerBW.png",0,0,125,125);
var attrsToAnimate = { rotation: "720" };
spinner.animate(attrsToAnimate, 60000);
```

非常好！我们现在有了旋转进度指示器，在加载地图时，可以让我们的用户知道加载的进度。但现在仍有一个问题：在地图加载完成后，它仍保留在原处，我们可以通过在现有的 `displayOnMap` 函数的开始处添加一行代码来解决此问题：

js/geolocation.js (excerpt)

```
function displayOnMap(position){
    document.getElementById("spinner").style.visibility = "hidden";
```

此行代码设置旋转器元素属性 `visibility` 为 `hidden`，它将有效地隐藏旋转器的 `div` 和我们加载到它上面的 SVG 图像。

11.2.5 画布与 SVG

现在，我们已经学习了画布和 SVG，您可以问一下自己，应该使用哪个呢？答案是：取决于您正在做的事情。

画布和 SVG 都可以使您绘制自定义的图形、路径及字体。但是，它们的独特之处分别是什么呢？

画布允许像素处理，正像我们在将彩色视频转换为黑白视频所看到的那样。画布的一个不足之处是：在立即模式下运行。也就是说，如果您想在画布上添加一些

东西，不能简单地在原有基础上进行添加。每一次更改画布上的内容时，都要从头绘制所有的东西。而且也不能通过 DOM 访问在画布上绘制的内容。然而，画布允许您将所创建的图像存储为 PNG 或 JPEG 文件格式。

相反，可以 DOM 访问在 SVG 中所绘制的内容，这是因为它的模式是保留模式——图像结构被保存在描述它的 XML 文档中。在此时，SVG 也有更多整套工具帮助您，如 Raphaël 库和 Inkscape。然而，SVG 是一个文件格式——而不是能够使您在平面上动态绘制的一套方法——您不能像在画布上处理像素那样处理 SVG 图像。例如，使用 SVG 将彩色视频转换为黑白色，就像我们使用画布那样，这将是不可可能的。

如果您需要将像素绘制到屏幕上，并且也不顾虑恢复和修改图形的功能，画布可能是更好的选择。另一方面，如果您需要访问或更改图片的某些具体方面，SVG 也许更适合。

如果两种技术都不适合静态图像，就一文不值了——至少在浏览器支持问题仍是一个绊脚石的一段时间。在本章中，我们在讲解画布和 SVG 时，使用了大量的静态图像的示例。这样很好地展示了它们的功能。但是在现实中，它们仅适用的情形是用户交互定义所要绘制的内容。

11.3 拖放

为了在我们网站上添加最后一个动态效果，我们将研究新的拖放 API。这个 API 允许我们规定某些元素是可以拖动的，然后规定将这些可拖动元素拖过或放到页面其他元素时所发生的状况。

以下浏览器支持拖放：

- 📱 Safari 3.2 及更高版本；
- 📱 Chrome 6.0 及更高版本；
- 📱 Firefox 3.5 及更高版本（在 Firefox 3.0 中，支持上一代的 API）；
- 📱 Internet Explorer 7.0 及更高版本；
- 📱 Android 2.1 及更高版本。

目前,在 Opera 中不支持 Drop and Opera。由于 Apple 更希望指导用户使用 DOM Touch API¹,因此在 iOS 中的设计不支持此 API。

使用拖放,主要可实现两种功能:从您的计算机上将文件拖到网页上(与 File API 的结合),或将元素拖入同一页面的其他元素中。在本章中,我们将重点介绍后者。



拖放和文件 API

如果您希望了解关于如何结合拖放和文件 API 的更多信息,以使用户能够从桌面将文件拖到网站上,可以在 Mozilla Developer Network²中找到一个非常好的指南。

目前仅 Firefox 3.6 及更高版本和 Chrome 支持文件 API。

将拖放添加到页面上,需要以下几个步骤。

- (1) 设置您想拖动的任何 HTML 元素的 draggable 属性。
- (2) 给任何可拖动的 HTML 元素的 dragstart 事件添加事件侦听器。
- (3) 给任何接受拖放的元素的 dragover 和 drop 事件添加事件侦听器。

11.3.1 给 WAI-ARIA 猫喂食

为了给页面增添一点乐趣,让我们添加一些小鼠的图像,所以在我们将这些图像拖到小猫图像上面的时候,能观察到小猫对它们的反应。在您开始担忧(或致电防止虐待动物协会)前,请放心,当然,那只是一个计算机鼠标。我们为鼠标使用一张来自 OpenClipArt 的图像³。

第一步是将这些新图像添加到我们的 `index.html` 文件。我们也分配给每一个鼠标图片一个 id:

```
js/dragDrop.js (excerpt)

<article id="ac3">
  <hgroup>
    <h1>Wai-Aria? HAHA!</h1>
```

¹ <http://developer.apple.com/library/safari/#documentation/AppleApplications/Reference/SafariWeb-Content/HandlingEvents/HandlingEvents.html>

² https://developer.mozilla.org/en/Using_files_from_web_applications

³ <http://www.openclipart.org/detail/111289>

```

    <h2>Form Accessibility</h2>
  </hgroup>

  <div class="content">
    <p id="mouseContainer">
      
      
      
    </p>
    :
  </div>

```

图 11.17 显示了起始状态的图片。



图 11.17 三只小鼠标，准备给 WAI-ARIA 猫喂食

11.3.2 使元素可拖动

下一步是使图像能够被拖动。要做到这一点，我们需要添加 `draggable` 属性，并将它的值设置为 `true`：

js/dragDrop.js (excerpt)

```





```




必须设置 draggable

请注意: draggable 不是一个布尔属性; 您必须将它明确地设置为 true。

现在, 我们已经将 draggable 设置为 true, 还需要为每张图像的 dragstart 事件设置一个事件侦听器。我们将使用 jQuery 的 bind 方法安装事件侦听器:

js/dragDrop.js (excerpt)

```
$( 'document' ).ready( function() {
    $( '#mouseContainer img' ).bind( 'dragstart', function( event ) {
        // handle the dragstart event
    } );
});
```

11.3.3 DataTransfer 对象

DataTransfer 对象拖放 API 是扼要介绍的新对象之一。这些对象允许我们设置和获取被拖动对象的数据。具体来说, DataTransfer 让我们可以定义以下两条信息。

- (1) 可拖动元素的存储数据类型。
- (2) 数据值本身。

在本例中, 我们要能够存储可拖动鼠标图像的 id, 这样我们就知道哪一个图像被拖动了。

为了做到这一点, 首先需要告诉 DataTransfer, 我们要通过在字符串 text/plain 传递存储一些纯文本, 然后将鼠标图像的 id 分配给它:

js/dragDrop.js (excerpt)

```
$( '#mouseContainer img' ).bind( 'dragstart', function( event ) {
    event.originalEvent.dataTransfer.setData( "text/plain",
    ↪ event.target.getAttribute( 'id' ) );
});
```

当拖动一个元素时, 我们将在 DataTransfer 对象中存储此元素的 id, 一旦放下该元素, 就将再次使用它。dragstart 事件的目标属性将是被拖动

的元素。



DataTransfer 和 jQuery

jQuery 库的 Event 对象仅可以使您能够访问它所知道的属性。这样，在您使用新的原生事件（如：DataTransfer）时将会产生一些问题，试图访问 jQuery 事件的 dataTransfer 属性将会导致一个错误。然后，您可以通过调用 jQuery 事件的 originalEvent 方法检索原始的 DOM 方法，就像我们上面所做的。这样可以使您能够访问任何浏览器所支持的属性——在本例中，便包括了新的 DataTransfer 对象。

当然，如果您从头就一直运行 JavaScript，这不是问题。

11.3.4 接受可以放下的元素

现在已将鼠标元素设置为可拖动的。然而，当您试图将它们拖动但不能放到任何地方时，这样就毫无乐趣可言了。

原因是，在默认情况下，页面上的元素并没有设置为接受拖动项目。为了覆盖特定元素的默认行为，我们必须停止它发生默认行为。可以通过创建两个事件侦听器做到这点。

我们需要监视的两个事件是 dragover 和 drop。正如您所期望的，当将某些东西拖动到一个元素上面时，便触发了 dragover，当您将它放到它上面时，便触发了 drop。

我需要防止这两个事件的默认行为——默认情况将禁止放下任何元素。

让我们在小猫图像上添加 id，使我们能够给它捆绑事件处理程序：

js/dragDrop.js (excerpt)

```
<article id="ac3">
  <hgroup>
    <h1>Wai-Aria? HAHA!</h1>
    <h2 id="catHeading">Form Accessibility</h2>
  </hgroup>

  
```

您也许已经注意到，我们也赋给了 h2 元素一个 id。这就是为什么在将鼠标图

像放到小猫上面时能够改变文本的原因。

现在，让我们处理 `dragover` 事件：

js/dragDrop.js (excerpt)

```
$('#cat').bind('dragover', function(event) {
    event.preventDefault();
});
```

这非常简单！在本例中，我们仅确保鼠标图片能够确实被拖到小猫图片的上面。我们还需要防止默认行为——使用 jQuery 的 `preventDefault` 方法可以达到此目的。

`drop` 处理程序的代码有些复杂，所以让我们一段一段地查看。我们的第一个任务是弄清在鼠标图片放在小猫的上面时，小猫说什么。为了显示这些，我们可以从 `DataTransfer` 对象检索放下的鼠标图像的 `id`，对于每个鼠标图片，不管将它们放下的顺序如何，我们都使用不同的短语。我们已经给出了 3 个小猫适当的选项：“MEOW!”、“Purr ...”和“NOMNOMNOM”。

我们将这些选项存储在名为 `mouseHash` 的对象中。第一步是声明对象：

js/dragDrop.js (excerpt)

```
$('#cat').bind('drop', function(event) {
    var mouseHash = {};
```

接下来，我们将利用 JavaScript 的对象，将键值对存储在里面，并且也存储在 `mouseHash` 对象的每一次响应以及与每个鼠标图像 `id` 相关的响应：

js/dragDrop.js (excerpt)

```
$('#cat').bind('drop', function(event) {
    var mouseHash = {};
    mouseHash['mouse1'] = "NOMNOMNOM";
    mouseHash['mouse2'] = "MEOW!";
    mouseHash['mouse3'] = "Purr...";
```

下一步是抓取 `h2` 元素，我们将更改此元素以反映小猫的响应：

js/dragDrop.js (excerpt)

```
var catHeading = document.getElementById('catHeading');
```

还记得我们使用 `setData` 将拖动的元素的 `id` 存储在 `DataTransfer` 对象中

吧？好，现在我们将检索它的 id。如果您已经猜到我们将需要一个名为 `getData` 的方法，那么，我想您猜对了：

js/dragDrop.js (excerpt)

```
var item = event.originalEvent.dataTransfer.getData("text/plain");
```

注意，我们已经将鼠标图像的 id 存储在名为 `item` 的变量中了。现在我们知道放下了哪个鼠标的图像了，并且有了标题，我们仅需要根据适当的响应更改文本：

js/dragDrop.js (excerpt)

```
catHeading.innerHTML = mouseHash[item];
```

我们使用存储在变量 `item`（拖动的鼠标图像的 id）中的信息为 `h2` 元素检索正确的信息。例如，拖动的鼠标图像是 `mouse1`，调用 `mouseHash[item]` 将检索“NOMNOMNOM”并将它设为 `h2` 元素的文本。

鉴于小鼠被“吃掉”，所以要从页面将它删除：

js/dragDrop.js (excerpt)

```
var mousey = document.getElementById(item);
mousey.parentNode.removeChild(mousey);
```

最后，但并非不重要，我们也必须防止默认行为不允许将元素放到小猫图像上面，和以前一样：

js/dragDrop.js (excerpt)

```
event.preventDefault();
```

图 11.18 显示了我们的快乐小猫还有一只小鼠没吃。

NOMNOMNOM



图 11.18 小猫已经吃掉两只小鼠

11.3.5 其他阅读材料

我们仅介绍了拖放 API 的基础部分，让您能够体会它的可用性。我们已经展示了如何使用 `DataTransfer` 将数据从拖动项目传递到它们放下的目标。您将使用这种功能去做什么完全取决于您自己！

如果您要了解关于拖放 API 的更多信息，有以下资料：

- Mozilla Developer Center 的拖放文档¹；
- W3C 的拖放规范²。

11.4 结束了，朋友们！

随着最后一些互动结束，关于 The HTML5 Herald 的工作接近尾声了，您在 HTML5 和 CSS3 世界的旅程已经开始！我们已经尽可能多地向您介绍了如今浏览器中的新功能，希望能够为您打下扎实的基础——至于如何使用它们，完全取决于您。

我们希望已经为您提供了一个关于在真正项目中如何使用这些功能的清晰画面。其中一些已经得到了很好的支持，并且浏览器将再一次快速发展，如果在使用时遇到一些元素还未得到支持，您可以在线寻求那些天才开发者的支持。通过这些志同道合的人们不断地努力，提供更多的后备程序及 `polyfill`，帮助我们推动和建立下一代网站和应用程序。

¹ https://developer.mozilla.org/En/DragDrop/Drag_and_Drop

² <http://dev.w3.org/html5/spec/dnd.html>



Modernizr

Modernizr 是一个开源 JavaScript 库，可以用来在用户的浏览器中测试 HTML5 的各项功能。它不是仅针对特定浏览器进行测试并做出结论，而是尝试根据下列事实制定决策，即 Modernizr 支持我们询问具体问题，比如“此浏览器是否支持地理定位？”，并明确地回答“是”或“否”。

使用 Modernizr 的第一步是从 Modernizr 网站下载它，网址为 <http://modernizr.com>。

拥有了脚本的副本后，您需要将脚本文件包含在您的页面中。在本例中，我们将脚本添加到了 head 中：

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>My Beautiful Sample Page</title>
  <script src="modernizr-1.7.min.js"></script>
</head>
```

您可以两种方式使用 Modernizr：与 CSS 一起使用，以及与 JavaScript 一起使用。

结合使用 Modernizr 和 CSS

当运行 Modernizr 时,它将会为检测到的每个功能在 HTML `<html>` 标签的 `class` 属性中添加一个条目,并且在浏览器不支持该功能的情况下,会添加前缀 `no-`。

例如,如果您正在使用 Safari 5,它几乎支持 HTML5 和 CSS3 的所有功能,则在运行 Modernizr 后,打开的 `<html>` 标签与下列内容类似:

```
<html class=" js flexbox canvas canvastext no-webgl no-touch
geolocation postmessage websqldatabase no-indexeddb hashchange
history draganddrop websockets rgba hsla multiplebgs backgroundsize
borderimage borderradius boxshadow textshadow opacity cssanimations
csscolumns cssgradients cssreflections csstransforms csstransforms3d
csstransitions fontface video audio localstorage sessionstorage
webworkers applicationcache svg no-inlinesvg smil svgclippaths">
```

下面是 Modernizr 在 Firefox 4 中为 `<html>` 标签添加的内容:

```
<html class=" js flexbox canvas canvastext webgl no-touch
geolocation postmessage no-websqldatabase indexeddb hashchange
history draganddrop no-websockets rgba hsla multiplebgs
backgroundsize borderimage borderradius boxshadow textshadow opacity
no-cssanimations csscolumns cssgradients no-cssreflections
csstransforms no-csstransforms3d csstransitions fontface video audio
localstorage no-sessionstorage webworkers applicationcache svg
inlinesvg smil svgclippaths">
```

为更好地利用 Modernizr 的此功能,我们首先应在 HTML 源代码中为 `html` 元素添加 `class no-js`。

```
<html class="no-js">
```

为什么要这样做?如果禁用了 JavaScript,那么 Modernizr 根本无法运行,但是如果启用了 JavaScript,Modernizr 做的第一件事就是将 `no-js` 更改为 `js`,如前面的 Safari 5 和 Firefox 4 示例所示。这样,您的样式将与已存在的 JavaScript 的样式相同。

您可能会认为“这听起来很棒,但是我应该如何处理此信息呢?”我们可以做的是使用这些类提供两种类型的 CSS:针对支持某些功能的浏览器设置一种样式,而针对其他浏览器设置另一种样式。

由于是在 html 元素中设置这些类的，因此我们可以使用子选择器根据指定功能的支持来定位页面上的任意元素。

下面是一个示例.class 为 cssgradients 元素中 id 为 #ad2 的任意元素(换句话说，也就是在 Modernizr 检测到渐变支持时的 html 元素)将接受我们在这里指定的任何样式：

```
.cssgradients #ad2 {
  /* gradients are supported! Let's use some! */
  background-image:
    -moz-linear-gradient(0% 0% 270deg,
      rgba(0,0,0,0.4) 0,
      rgba(0,0,0,0) 37%,
      rgba(0,0,0,0) 83%,
      rgba(0,0,0,0.06) 92%,
      rgba(0,0,0,0) 98%);
  :
}
```

但是如果不支持 CSS 渐变呢？我们可以更改样式以使用简单的 PNG 背景图像，重新创建相同的渐变外观。下面介绍我们如何做到这一点：

```
.no-cssgradients #ad2 {
  background-image:
    url(../images/put_a_replacement_img_here.png)
}
```

我们可以使用 Modernizr 添加到 html 元素的类的另一种方式，即使用拖放。我们在第 11 章中介绍了拖放，其中，我们添加了几张电脑鼠标的图片，它们可以被拖动到我们的猫图片上，被猫吃掉。这些图像都存储在 id 为 mouseContainer 的 div 中。

但是，在 Opera 中，拖放将不起作用，但是为什么甚至不会显示鼠标图像呢？如果不支持拖放，那么我们可以使用 Modernizr 隐藏 div。

css/styles.css (excerpt)

```
.no-draganddrop #mouseContainer {
  visibility:hidden;
  height:0px;
}
```

如果支持拖放，我们可以将 div 中的所有内容水平对齐：

css/styles.css (excerpt)

```
.draganddrop #mouseContainer {
    text-align:center;
}
```

结合使用 Modernizr 和 JavaScript

如果访问者的浏览器缺少对您使用的任意 HTML5 元素的支持，我们可以在 JavaScript 中使用 Modernizr 来提供一些回退。

当 Modernizr 运行并将这些类全部添加到您的<html>元素时，它也会创建一个全局 JavaScript 对象，您可以使用该对象进行功能支持测试。该对象名为 Modernizr，它包含了每个 HTML5 功能都具有的一个属性。

下面是几个示例：

```
Modernizr.draganddrop;
Modernizr.geolocation;
Modernizr.textshadow;
```

每个属性或者是 true 或者是 false，取决于该功能在访问者的浏览器中是否可用。这很有用，因为我们询问问题，比如“访问者的浏览器是否支持地理定位？”，然后根据回答采取相应的操作。

下面是使用 Modernizr 的 if/else 代码块来测试地理定位支持的一个示例：

```
if (Modernizr.geolocation) {
    // go ahead and use the HTML5 geolocation API,
    // it's supported!
}
else {
    // There is no support for HTML5 geolocation.
    // We may try another library, like Google Gears
    // (http://gears.google.com/), to locate the user.
}
```

在 Internet Explorer 8 和早期版本中设置 HTML5 元素样式的支持

如第 2 章所述，Internet Explorer 8 和早期版本将禁止设置无法识别的元素的样

式。然后我们介绍了 Remy Sharp 提出的解决方案 HTML5 shiv，这个解决方案可以解决此问题。但是，我们也间接提到了，Modernizr 也可以为我们解决这个问题！

正如 Modernizr 的文档页面所述：Modernizr 在 JavaScript 中运行一个小循环，以支持在 Internet Explorer 中设置来自 HTML5 的各种元素的样式。注意，这并不是说它突然间使 Internet Explorer 支持音频或视频元素，而只是意味着您可以使用 section 代替 div，并在 CSS 中设置它们的样式。

换句话说，现在我们不再需要 HTML5 shiv 来在 Internet Explorer 8 和早期版本中设置新语义元素的样式。如果我们正在使用 Modernizr，那么它将为我们执行这一操作。



位置、位置、位置

如果您正在使用 Modernizr 而不是 HTML5 shiv，则将需要将其放在页面顶部。否则，在 IE 中您的元素将显示为未设置样式，除非浏览器已到达 Modernizr 脚本的位置并执行了该脚本。

其他阅读材料

要了解关于 Modernizr 的更多信息，请参见以下内容。

- Modernizr 文档：<http://www.modernizr.com/docs/>。
- 一个相当全面的 HTML5 和 CSS3 属性的 polyfills 最新列表，可以与 Modernizr 一起使用，在下列网站进行维护：<https://github.com/Modernizr/Modernizr/wiki/HTML5-Cross-browser-Polyfills>。
- A List Apart 文章 Taking Advantage of HTML5 and CSS3 with Modernizr，网址为 <http://www.alistapart.com/articles/taking-advantage-of-html5-and-css3-with-modernizr/>。

WAI-ARIA

在第 2 章和第 3 章中，我们介绍了大量基础知识，解释使用 HTML5 的新语义元素为页面添加更大的可访问性和可移植性的潜在好处。但是，仅凭改进的语义有时并不能够使成熟的 Web 应用程序完全可以访问。

为了使我们的用户能够尽可能地访问页面的内容和功能，我们需要增强 WAI-ARIA 提供的功能，并扩展 HTML5（或者任何标记语言）已有的功能。

我们在这里不会深入探讨 WAI-ARIA，介绍它可能需要很多章节，但是我们认为在这里提一下它很重要，以便您了解自己的选择。

WAI-ARIA 表示 Web Accessibility Initiative-Accessible Rich Internet Applications，即无障碍富互联网应用程序。W3C 网站上 WAI-ARIA 的概述为¹：

[...]一种使残疾人能够更容易访问 Web 内容和 Web 应用程序的方式。这特别有助于使用 Ajax、HTML、JavaScript 和相关技术开发的动态内容和高级用户界面控件。

依赖屏幕阅读器技术的用户，或者是无法使用鼠标的用户，通常无法使用某些网站和 Web 应用程序的功能，例如，滑块、进度条和选项卡式界面。使用 WAI-ARIA，即使内容和功能包含在复杂的应用程序架构中，您也能够克服页面的这些缺点。因此，

¹ <http://www.w3.org/WAI/intro/aria.php>

对于依赖辅助技术的用户来说，一些通常无法访问的网站内容现在就可以访问了。

WAI-ARIA 如何补充语义

WAI-ARIA 向元素分配角色，并指定角色属性和状态。下面是一个简单的示例：

```
<li role="menuitemcheckbox" aria-checked="true">Sort by Date</li>
```

应用程序可能会将列表项用作一个链接元素来对内容进行分类，但是，如果没有 `role` 和 `aria-checked` 属性，屏幕阅读器将无法确定该元素的目的。单独的语义（在本例中是列表项）不会说明任何问题。通过添加这些属性，辅助设备能更好地理解此函数的目的。

对于一些语义元素，例如，`header`、`h1` 和 `nav`，WAI-ARIA 属性不是必需的，因为这些元素已经说明了它们是什么。相反，它们应该用于从元素本身看不出其功能和用途的元素。

WAI-ARIA 的现状

与 HTML5 一样，WAI-ARIA 规范是新的，因此这些技术还没有提供我们想要的所有好处。尽管我们介绍了 WAI-ARIA 扩展页面元素的语义的方式，但是可能有必要在已表明其意思的元素上包含 WAI-ARIA 角色，因为辅助技术还不能支持所有新的 HTML5 语义。换句话说，WAI-ARIA 可作为一种权宜之计，在屏幕阅读器不可用的时候提供对到 HTML5 页面的访问。

让我们看一个网站导航，例如：

```
<nav>
  <ul role="navigation">
    :
  </ul>
</nav>
```

这里似乎是正在翻倍：`nav` 元素暗示其包含的链接列表组成了一个导航控件，但是我们仍然为列表添加了 WAI-ARIA 角色 `navigation`。因为 WAI-ARIA 和 HTML5 是新技术，所以这种类型的翻倍通常是必要的：一些缺少 HTML5 支持的浏览器和屏幕阅读器将能够支持 WAI-ARIA，反之亦然。

这是否意味着全面支持 HTML5 后 WAI-ARIA 将会变成多余的？不是，WAI-ARIA 中有不对应于 HTML5 元素的角色，例如，timer¹角色。尽管您可以使用 HTML5 的 time 元素表示一个计时器，然后再使用 JavaScript 更新它，但是却无法告知屏幕阅读器它是一个计时器而不是静态时间表示。

为了让屏幕阅读器访问 WAI-ARIA 角色，浏览器必须通过一个可访问性 API 来公开它们。这允许屏幕阅读器以类似于访问本地桌面控件的方式与元素进行交互。

2010 年底在 A List Apart 上发表的一篇文章中，以下列内容概述了浏览器中的 WAI-ARIA 支持和辅助设备²：

在最新的浏览器和屏幕阅读器版本中，对 WAI-ARIA [...] 一些内容的支持已经相当好。但是，仍然存在许多问题。

最后，值得注意的是，并不是所有可以从 WAI-ARIA 角色受益的用户都正在使用这些角色。2010 年 12 月，名为 WebAIM (Web Accessibility In Mind) 的组织进行了一个第三方屏幕阅读器用户调查³，调查表明超过 50% 的参与者没有使用 WAI-ARIA 功能，也不知道存在这种功能。

简而言之，有一些支持 WAI-ARIA，并且通过包含这些属性不会损坏 HTML5 文档，因为它们在 HTML5 中有效。虽然全部受益仍有待于观察，但是它们肯定会随时间而增加。

其他阅读材料

如前所述，本书不会介绍所有 WAI-ARIA 角色，但是如果您想了解更多内容，我们强烈推荐官方规范⁴。W3C 包含了一个简单的入门介绍⁵和一个创作实践指南⁶。

¹ <http://www.w3.org/TR/wai-aria/roles#timer>

² <http://www.alistapart.com/articles/the-accessibility-of-wai-aria/>

³ <http://webaim.org/projects/screenreadersurvey3/>

⁴ <http://www.w3.org/TR/wai-aria/>

⁵ <http://www.w3.org/TR/wai-aria-primer/>

⁶ <http://www.w3.org/TR/wai-aria-practices/>



微数据

微数据是得到快速采用和支持的另一项技术，但与 WAI-ARIA 不同，它实际上是 HTML5 的一部分。微数据规范¹仍处于开发初期，但此技术值得在此一提，因为它可使我们了解文档可读性和语义的未来。

在规范中，微数据被定义为一种机制，“允许以一种易写的方式使用明确的解析模型将机器可读的数据嵌入 HTML 文档。”

使用微数据，页面作者可以将具体标签添加到 HTML 元素以注释它们，以便机器或机器人能够读取它们。这可以通过一种自定义的术语来完成。例如，您可能想要一个脚本或其他第三方服务能够访问页面并与页面上的具体元素以某种方式进行交互。使用微数据，您可以扩展现有语义（比如 `article` 和 `figure`），以允许这些服务可以专门访问注释的内容。

这听起来可能会让人感到困惑，所以我们来看一个实际示例。假设您的网站包含电影评论。您可能会将每条评论放在 `article` 元素中，使用几个星号或百分比分数对每条评论进行评分。但是，当机器遇到这种情况时，比如 Google 的搜索蜘蛛，则无法得知内容的哪部分是实际评论，在页面上所看到的只是一堆文字。

为什么机器想要了解您关于电影的看法？值得考虑一下的是，Google 最近开始在其搜索结果页面中显示更加丰富的信息，以为搜索者提供不仅仅是查询文本匹配

¹ <http://www.w3.org/TR/microdata/>

的内容。这是通过使用微数据或其他类似的技术来阅读编码到这些网站页面中的评论信息来实现的。电影评论信息的另一个示例如图 C.1 所示。

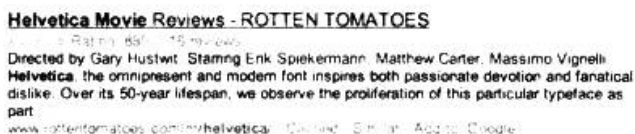


图 C.1 Google 利用微数据在搜索结果中显示更多信息

通过使用微数据，您可以精确指定页面的哪部分与评论、人员和活动等相对应，以软件应用程序可以理解并利用的一致词汇来指定。

HTML5 的语义还不够吗

您可能会想，如果使用现有的 HTML 时某个元素不可用，那么它能有什么作用？毕竟，现在 HTML5 规范包含一些新元素，支持更明确的标记。但是 HTML5 的创造者非常谨慎，以确保作为 HTML5 规范一部分的元素是常用的元素。

为 HTML 添加一些仅有少数人使用的元素可能会事与愿违。这会不必要地增加语言，使其从规范作者或标准本身来看都难以进行维护。

另一方面，微数据支持您为每种具体情形创建自己的自定义词汇，这些情形使用 HTML5 的语义元素是不可能的。因此，现有的 HTML 元素和添加到 HTML5 的新元素都作为一种语义标准，而开发人员可以针对其特定需要创建具体的注释。

微数据语法

微数据适用于现有的、组织良好的 HTML 内容，并通过名称/值对（也称为属性）添加到文档中。微数据不允许您创建新元素，而是为您提供添加自定义属性以扩展现有元素语义的选项。

下面是一个简单示例：

```
<aside itemscope>
  <h1 itemprop="name">John Doe</h1>
  <p></p>
  <p><a href="http://www.sitepoint.com" itemprop="url">Author's
  ↪website</a></p>
</aside>
```


在上面的示例中，我们有一个常规作者简介，放置在 `aside` 元素中。您发现的第一不同之处是 `itemscope` 属性。它将 `aside` 元素看做一个容器，定义我们的微数据词汇的范围。`itemscope` 属性则定义规范称之为项的内容。每个项都有一组名称/值对。

定义我们的词汇范围的能力支持我们在单个页面定义多个词汇。在上面的示例中，`aside` 元素中的所有名称/值对都是微数据词汇的一部分。

在 `itemscope` 属性后面，是另一个令人感兴趣的项 `itemprop` 属性，它的值为 `name`。此时，解释脚本如何从这些属性获取信息以及名称/值对的含义可能是个好主意。

了解名称/值对

名称是一个利用 `itemprop` 属性定义的属性。在我们的示例中，第一个属性名称碰巧为 `name`。在此范围中还有两个属性名称：`photo` 和 `url`。

给定属性的值定义是不同的，这取决于声明该属性的元素。对于大多数元素，值来自其文本内容。例如，在我们的示例中，`name` 属性会从 `<h1>` 的开始标签和结束标签之间的文本内容获取其值，而其他元素要区别对待。

`photo` 属性从图像的 `src` 属性获取其值，因此该值由一个指向作者照片的 URL 组成。尽管 `url` 属性是在具有文本内容（即短语 `Author's website`）的元素上定义的，但是它不使用此文本内容来确定其值，而是从 `href` 属性获取其值。

其他不会使用其相关文本内容来定义微数据值的元素包括 `meta`、`iframe`、`object`、`audio`、`link` 和 `time`。关于从文本内容之外的位置获取值的元素的完整列表，请参见微数据规范的“值”部分¹。

微数据命名空间

目前为止，我们所介绍的是可接受的微数据，它不会进行重用，但是这有些不切实际。如前所述，微数据的实际能力只有在第三方脚本和页面作者可以访问我们的名称/值对并为它们寻找有益用途时才会释放出来。

¹ <http://www.w3.org/TR/microdata/#values>

为了释放微数据的实际能力，必须通过 `itemtype` 属性来定义每个项的类型。记住，微数据中的项指的是具有 `itemtypescope` 属性集的元素。每个元素和该元素中的名称/值对是项的一部分。因此，`itemtype` 属性的值定义项的词汇的命名空间。为我们的示例添加一个 `itemtype`：

```
<aside itemscope itemtype="http://www.data-vocabulary.org/Person">
  <h1 itemprop="name">John Doe</h1>
  <p></p>
  <p><a href="http://www.sitepoint.com" itemprop="url">Author's
  ➤website</a></p>
</aside>
```

在我们的项中，我们使用 URL `http://www.data-vocabulary.org/`，这是 Google 的一个域。Google 有很多微数据词汇，包括 `Organization`、`Person`、`Review` 和 `Breadcrumb` 等。

其他阅读材料

本章对微数据的简单介绍几乎没有对它进行公正的评论，但是我们希望您能了解在使用此技术扩展文档的语义时能够进行的操作。

微数据是一个非常广泛的话题，除了本章外，您还需要阅读和研究其他资源。考虑到这一点，如果您想深入探索微数据提供的可能性，下面列出了一些链接可供参考：

- HTML5 Doctor 上的 Extending HTML5 – Microdata¹；
- W3C 微数据规范²；
- Mark Pilgrim 对微数据的绝佳概述³；
- Google 的 Rich Snippets Help⁴。

¹ <http://html5doctor.com/microdata/>

² <http://www.w3.org/TR/microdata/>

³ <http://diveintohtml5.org/extensibility.html>

⁴ <http://www.google.com/support/webmasters/bin/answer.py?hl=en&answer=99170>